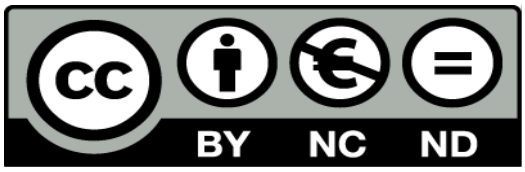
UNIVERSIDAD
DE LA RIOJA

TRABAJO FIN DE GRADO

Título
Diseño y desarrollo de un gestor de “Catalogo de servicios”
Autor/es
María Ibeas Ochoa
Director/es
Juan José Olarte Larrea
Facultad
Facultad de Ciencias, Estudios Agroalimentarios e Informática
Titulación
Grado en Ingeniería Informática
Departamento
Curso Académico
2012-2013



Diseño y desarrollo de un gestor de "Catalogo de servicios", trabajo fin de grado de María Ibeas Ochoa, dirigido por Juan José Olarte Larrea (publicado por la Universidad de La Rioja), se difunde bajo una Licencia
Creative Commons Reconocimiento-NoComercial-SinObraDerivada 3.0 Unported.
Permisos que vayan más allá de lo cubierto por esta licencia pueden solicitarse a los titulares del copyright.

© El autor
© Universidad de La Rioja, Servicio de Publicaciones, 2013
publicaciones.unirioja.es
E-mail: publicaciones@unirioja.es

Contenido

1. RESUMEN 5
2. RESUMEN INGLÉS 5
3. DOCUMENTO DE OBJETIVOS DEL PROYECTO 6
3.1 Objetivos 6
3.2 Antecedentes 6
3.3 Alcance 6
3.4 Metodología 6
3.5 Glosario 7
3.6 Requisitos iniciales 7
3.7 Las fases 8
3.8 Entregables 8
3.9 Estructura de descomposición de tareas 9
3.10 Planificación detallada 9
3.11 Estimaciones globales 11
3.12 Horario de trabajo 11
3.13 Diagrama de Gantt 12
3.14 Riesgos 12
4. FASE 1 14
4.1 Análisis 14
4.1.1 Definición de la fase 1. 14
4.1.2 Actores del sistema. 14
4.1.3 Análisis de requisitos. 14
4.1.4 Identificación de los casos de uso. 14
4.1.5 Especificación de los casos de uso. 15
4.1.6 Diagrama de clases. 16
4.2 Diseño 17
4.2.1 Arquitectura del sistema. 17
4.2.2 Diseño de la Base de datos 18
4.2.3 Diseño de las clases. 19

4.2.4 Diseño de interfaces. 21
4.3 Construcción 22
4.3.1 Implementación 22
4.3.2 Pruebas 26
4.3.3 Conclusiones 27
5. FASE 2 28
5.1 Análisis 28
5.1.1 Definición de la fase 2. 28
5.1.2 Actores del sistema. 28
5.1.3 Análisis de requisitos. 28
5.1.4 Identificación de los casos de uso. 28
5.1.5 Especificación de los casos de uso. 29
5.1.6 Diagrama de clases. 31
5.2 Diseño 31
5.2.1 Diseño de la Base de datos 31
5.2.2 Diseño de las clases. 31
5.2.3 Diseño de interfaces. 33
5.3 Construcción 34
5.3.1 Implementación 34
5.3.2 Pruebas 37
5.3.3 Conclusiones 38
6. FASE 3 39
6.1 Análisis 39
6.1.1 Definición de la fase 3. 39
6.1.2 Actores del sistema. 39
6.1.3 Análisis de requisitos. 39
6.1.4 Identificación de los casos de uso. 39
6.1.5 Especificación de los casos de uso. 40
6.1.6 Diagrama de clases 41

6.2 Diseño 42
6.2.1 Diseño de la base de datos. 42
6.2.2 Diseño de las clases. 42
6.2.3 Diseño de interfaces. 44
6.3 Construcción 45
6.3.1 Implementación 45
6.3.2 Pruebas 47
6.3.3 Despliegue 48
7. CONCLUSIONES 49

1. RESUMEN

La aplicación desarrollada en este Trabajo Fin de Grado constituye una herramienta informática que permitirá la consulta de todos los servicios que ofrece la Universidad de la Rioja recopilados en un catálogo. Este catálogo será accesible desde la página Web del Servicio Informático. En él aparecerán aplicaciones, documentos y enlaces a páginas que ofrezcan servicios tanto para estudiantes, PDI y PAS. La información estará ordenada en diferentes niveles y subniveles para facilitar la búsqueda de los distintos servicios y además proporcionará un buscador para poder encontrar cualquiera de ellos mediante palabras clave que lo definan.

Los servicios llevarán asociados una restricción por acceso de tal forma que podrán existir servicios accesibles únicamente por determinados usuarios.

2. RESUMEN INGLÉS

The application developed in this Final Project is a tool that allows the query of all services offered by the University of La Rioja collected in a catalog. This catalog will be accessible from the Computer Service website. There will be applications, documents and links to pages that offer services for both students, PDI and PAS. The information is sorted into different levels and sublevels to make easier the search for the distinct services and provide also a search engine to find any of them using keywords that define it.

Some services are limited by access restriction so they can be accessed only by certain users.

3. DOCUMENTO DE OBJETIVOS DEL PROYECTO

3.1 Objetivos

El objetivo fundamental de este TFG es la realización de una aplicación Web que ofrezca una relación de todos los servicios que ofrece la Universidad para que sea accesible desde la página Web del Servicio Informático.

Esta aplicación podrá ser usada por todos los usuarios de la Universidad, pudiendo variar el contenido del catálogo de productos ofrecido dependiendo del tipo de usuario que acceda a la aplicación como veremos más adelante.

3.2 Antecedentes

No existe actualmente una página donde se recojan todos los servicios que ofrece la Universidad. La Universidad de La Rioja ofrece ciertos recursos o aplicaciones que representan los distintos servicios que se mostrarán en el catálogo. Algunos de estos servicios se pueden encontrar en la página de la Universidad pero existen varios problemas:

- Los servicios no están centralizados.
- La información relativa a cada servicio no es suficientemente clara.
- No existe el control de acceso a los diferentes servicios.

3.3 Alcance

Se han establecido una serie de requisitos mínimos que deben alcanzarse:

- Gestión de categorías.
- Gestión de servicios.
- Web del catálogo de servicios.
- Gestión de relaciones entre categorías.
- Control de acceso.

3.4 Metodología

La metodología a seguir en el desarrollo del proyecto será la del Proceso Unificado del Desarrollo de Software siguiendo un ciclo de carácter iterativo e incremental.

El proyecto está dividido en iteraciones y cada iteración se divide en una serie de disciplinas: Análisis de requisitos, Diseño, Implementación y Pruebas.

Utilizaremos un ciclo de vida iterativo e incremental porque nos permitirá mostrar al cliente partes de la aplicación plenamente operativos en etapas tempranas del desarrollo. De esta forma podremos tener una evaluación del cliente y disminuiremos los riesgos de que al finalizar la aplicación no satisfaga al cliente.

3.5 Glosario

Perfiles de la universidad que se van a utilizar en el control de acceso de la aplicación:

- **ANOF:** Alumnos con estudios NO Oficiales.
- **AUR:** Alumnos Aúreos.
- **CAC:** Usuarios de Campus Activo.
- **CAP:** Alumnos del Curso de Adaptación Pedagógica.
- **COFI:** Cuentas oficiales.
- **DIA:** Fundación DIALNET.
- **EAD:** Usuarios (docentes ó adm.) de escuelas adscritas.
- **EX_ANOF:** Ex alumno con estudios NO Oficiales.
- **EX_AUR:** Ex áureo.
- **EX_PAS:** Ex personal de Administración y Servicios.
- **EX_PDI:** Ex personal Docente e Investigador.
- **EX_PSC:** Ex alumnos (PSC) que ya no están matriculados.
- **EX_TCL:** Ex alumnos (TCL) que ya no están matriculados.
- **EXT:** Usuarios externos sin relación alguna con la UR.
- **FUR:** Personal de la Fundación de la Universidad de la Rioja.
- **G9:** Usuarios del G9.
- **GDI:** Usuarios para alta en la Gestión de Direcciones IP.
- **INV:** Personal de Investigación.
- **INVDEP:** Investigadores de Departamento.
- **ONL:** Alumno Online.
- **PAS:** Personal de Administración y Servicios.
- **PDI:** Personal Docente e Investigador.
- **PRE:** Alumnos registrados vía formulario para la preinscripción.
- **PSC:** Alumnos presenciales de primer y segundo ciclo.
- **PSC_RIP:** Ex alumnos (PSC) que ya no están matriculados.
- **REP:** Alumnos procedentes del reparto.
- **TCL:** Alumnos de tercer ciclo.
- **VISI:** Visitantes.

3.6 Requisitos iniciales

La aplicación deberá permitir:

1. Administración de categorías: mediante formularios se deberá permitir el alta, modificación y borrado de categorías, así como gestionar la dependencia de unos con respecto a otros.

2. Cada servicio estará formado por los siguientes atributos: descripción del servicio, persona encargada, datos de contacto, enlace al recurso o aplicación, información adicional o notas y documentos adjuntos que se necesiten subir.

3. Administración de servicios: se permitirá en este formulario el mantenimiento de los servicios, asignando dichos servicios a categorías. Esta asignación será la que posteriormente permitirá la agrupación en la consulta.

4. Presentación de los servicios: mediante este formulario se presentarán todos los servicios a la comunidad. Se presentarán todos los servicios agrupados según los distintos niveles. También se permitirá la búsqueda de aquellos servicios que contengan un texto en los atributos “descripción” y “nota”.

5. Administración de los permisos de acceso a los SERVICIOS de tal forma que un SERVICIO tendrá asociados los usuarios que pueden acceder, siendo estos usuarios los siguientes:

    a. PÚBLICO: este es un usuario especial e indica que el servicio puede ser consultado por cualquiera.

    b. Perfiles: como por ejemplo PSC, TCL, PAS… Indicará que cualquier persona que tenga ese perfil podrá acceder.

    c. Usuarios individuales.

    Por lo tanto a la hora de rellenar los servicios se deberán añadir también los usuarios de acceso.

    Cuando un SERVICIO no tenga una entrada con el acceso PÚBLICO habrá que añadirle la etiqueta “(restringido)” en la visualización y solicitarle el login y contraseña al seleccionarlo. De esta forma ya se podrá comprobar si pertenece a un perfil que tenga acceso o a un usuario autorizado.

3.7 Las fases

El proyecto se va a dividir en tres fases. Al finalizar cada una de estas fases se obtendrá un entregable de la aplicación que será mostrado al cliente. En cada una de estas fases se realizarán las siguientes tareas:

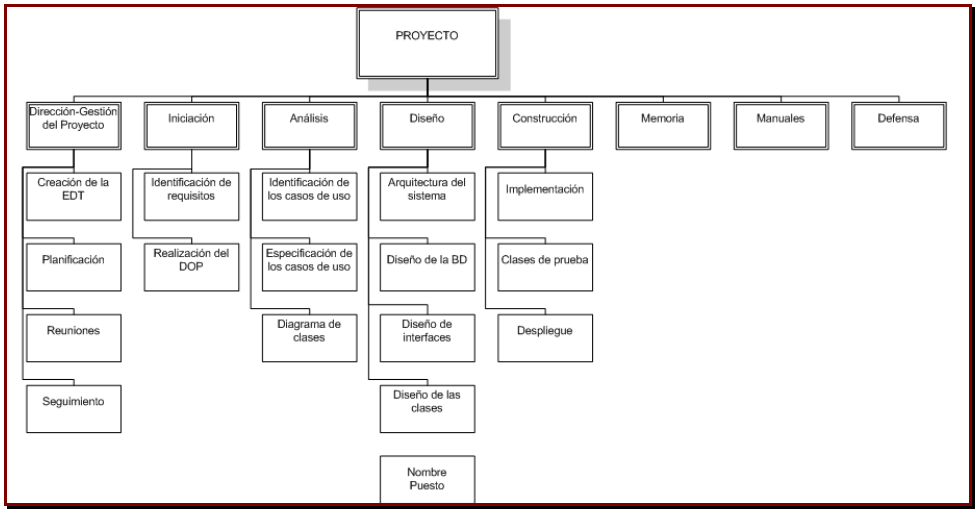
- Fase 1: formularios para crear tanto categorías como servicios y la posibilidad de ver todas las categorías y los servicios creados.
- Fase 2: formularios para modificar y borrar categorías y servicios y diagrama de la aplicación.
- Fase 3: control de acceso y búsqueda de categorías y servicios.

3.8 Entregables

El proyecto estará constituido por los siguientes documentos, cuya entrega será realizada a la finalización del mismo:

- Memoria
- Manual de usuario e instalación
- Aplicación

3.9 Estructura de descomposición de tareas



3.10 Planificación detallada

3.10.1 Dirección-Gestión del proyecto: 15 horas

En este apartado se agrupan las tareas de planificación y seguimiento del proyecto incluyendo también las tareas derivadas de las reuniones que se lleven a cabo.

- **Creación de la Estructura de Descomposición de Trabajo:** Realización de la estructura de las tareas que van a conformar el proyecto **(1 hora).**
- **Planificación:** Estimación temporal para realizar cada una de las tareas (**3 horas).**
- **Reuniones:** Corresponde con las tareas derivadas de las reuniones con el director del proyecto o con el servicio informático (**6 horas).**
- **Seguimiento:** Seguimiento de la duración de las tareas del proyecto y del desfase respecto a las estimaciones aquí propuestas (**5 horas).**

3.10.2 Iniciación: 14 horas

Corresponde con las tareas realizadas al comienzo del proyecto y sirven para realizar una introducción a la definición del problema.

- **Identificación de requisitos:** Tareas realizadas para la obtención de requisitos que serán detallados en el análisis **(5 horas)**.
- **Realización del DOP:** Creación de Documentos de Objetivos del Proyecto **(9 horas)**.

3.10.3 Análisis: 40 horas

En este apartado se realiza una especificación detallada del sistema.

- **Identificación casos de uso:** Definición, mediante diagramas, del papel que desempeñan los actores dentro del sistema permitiendo una visión sencilla de su comportamiento **(10 horas)**.
- **Especificación casos de uso:** Explicación detallada de cada uno de los casos de uso **(25 horas)**.
- **Diagrama de clases:** Definición, mediante un diagrama, de la estructura del sistema que permite observar sus clases, atributos y relaciones entre ellas **(5 horas).**

3.10.4 Diseño: 35 horas

- **Arquitectura del sistema:** Explicación de la arquitectura que se va a utilizar en la aplicación **(2 horas).**
- **Diseño de la BD:** Diseño conceptual del modelo de datos **(10 horas)**.
- **Diseño de las clases:** Cabeceras de los métodos que conforman las clases **(8 horas)**
- **Diseño de interfaces:** Realización de los prototipos de interfaz de la aplicación **(15 horas)**.

3.10.5 Construcción: 160 horas

En este apartado se desarrolla el sistema, generando el código necesario y comprobando que su funcionamiento sea correcto para su posterior implantación.

- **Implementación:** En este apartado se implementará la base de datos, la lógica de la aplicación y las interfaces **(130 horas).**
- **Pruebas:** Se desarrollarán las pruebas necesarias para comprobar el correcto funcionamiento de la aplicación **(20 horas).**
- **Despliegue:** Tareas necesarias para la instalación y configuración de la aplicación en un servidor **(10 horas).**

3.10.6 Memoria:

Creación de la memoria del Trabajo Fin de Grado **(15 horas).**

3.10.7 Manuales:

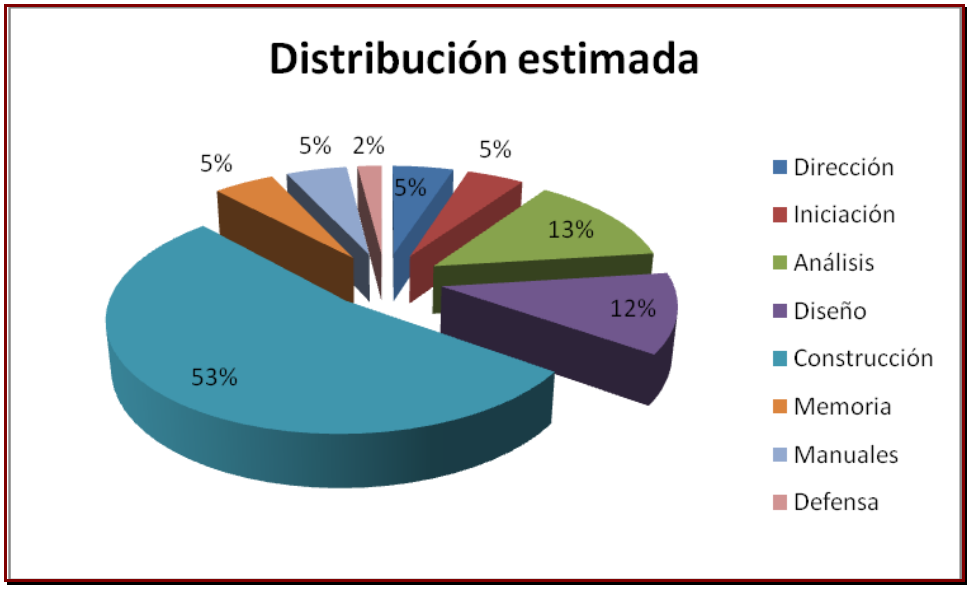
Creación del manual de usuario para el Servicio Informático **(15 horas).**

3.10.8 Defensa:

Tareas relacionadas con la preparación de la defensa del Trabajo Fin de Grado **(6 horas).**

3.10.9 Duración total: 300 horas.

3.11 Estimaciones globales



3.12 Horario de trabajo

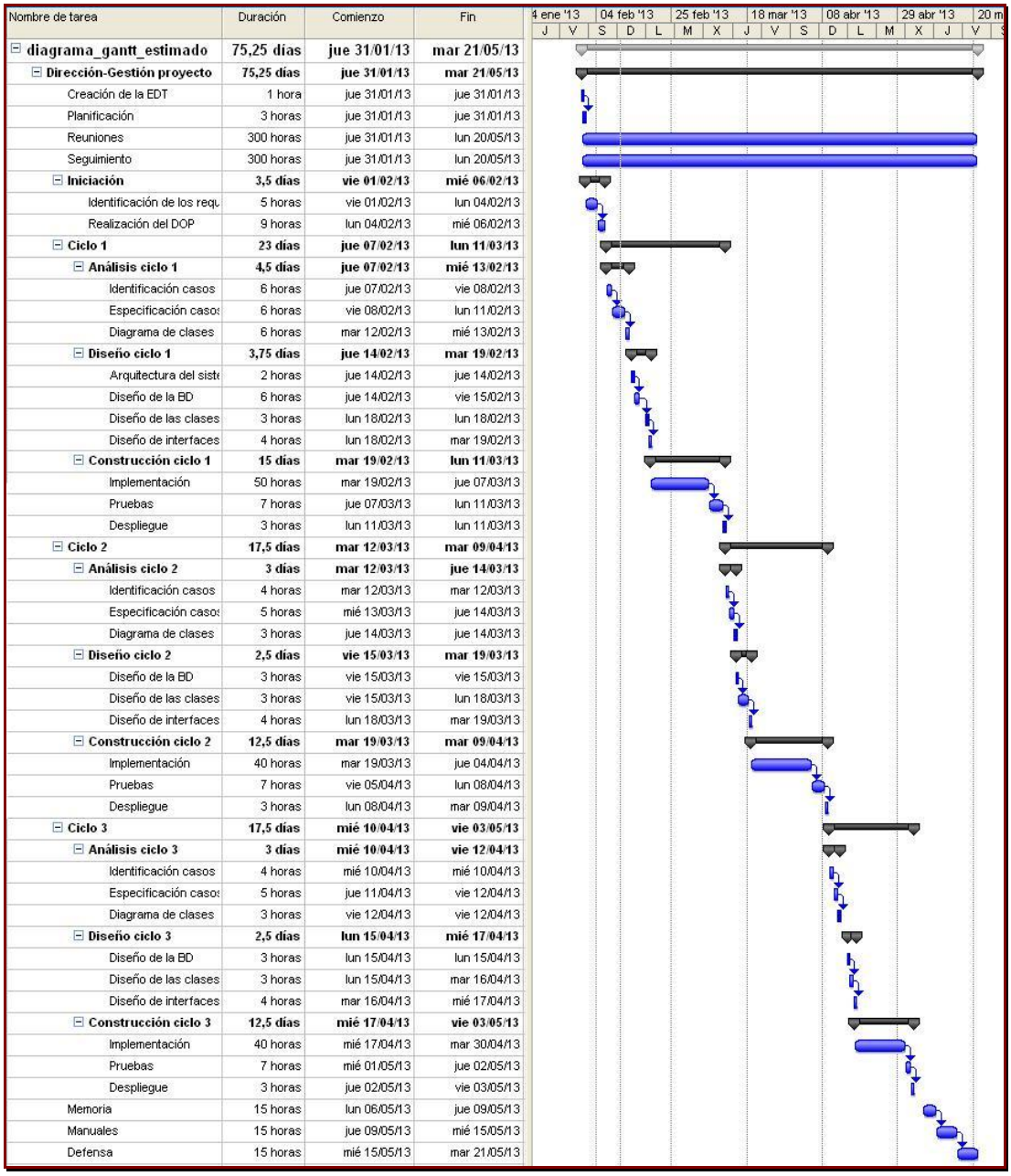
A continuación se muestra el horario que se va a dedicar al desarrollo del proyecto:

	Lunes	Martes	Miércoles	Jueves	Viernes
9 – 10					
10 – 11					
11 – 12					
12 – 13					
13 – 16					
16 – 17					
17 – 18					
18 – 19					
19 – 20					

Horas fijas: Destinadas a la realización del proyecto.

Horas adicionales: Destinadas a la recuperación de tiempos.

3.13 Diagrama de Gantt



Nombre de tarea	Duración	Comienzo	Fin
diagrama_gantt_estimado	75,25 días	jue 31/01/13	mar 21/05/13
Dirección-Gestión proyecto	75,25 días	jue 31/01/13	mar 21/05/13
Creación de la EDT	1 hora	jue 31/01/13	jue 31/01/13
Planificación	3 horas	jue 31/01/13	jue 31/01/13
Reuniones	300 horas	jue 31/01/13	lun 20/05/13
Seguimiento	300 horas	jue 31/01/13	lun 20/05/13
Iniciación	3,5 días	vie 01/02/13	mié 06/02/13
Identificación de los requ	5 horas	vie 01/02/13	lun 04/02/13
Realización del DOP	9 horas	lun 04/02/13	mié 06/02/13
Ciclo 1	23 días	jue 07/02/13	lun 11/03/13
Análisis ciclo 1	4,5 días	jue 07/02/13	mié 13/02/13
Identificación casos	6 horas	jue 07/02/13	vie 08/02/13
Especificación casos	6 horas	vie 08/02/13	lun 11/02/13
Diagrama de clases	6 horas	mar 12/02/13	mié 13/02/13
Diseño ciclo 1	3,75 días	jue 14/02/13	mar 19/02/13
Arquitectura del siste	2 horas	jue 14/02/13	jue 14/02/13
Diseño de la BD	6 horas	jue 14/02/13	vie 15/02/13
Diseño de las clases	3 horas	lun 18/02/13	lun 18/02/13
Diseño de interfaces	4 horas	lun 18/02/13	mar 19/02/13
Construcción ciclo 1	15 días	mar 19/02/13	lun 11/03/13
Implementación	50 horas	mar 19/02/13	jue 07/03/13
Pruebas	7 horas	jue 07/03/13	lun 11/03/13
Despliegue	3 horas	lun 11/03/13	lun 11/03/13
Ciclo 2	17,5 días	mar 12/03/13	mar 09/04/13
Análisis ciclo 2	3 días	mar 12/03/13	jue 14/03/13
Identificación casos	4 horas	mar 12/03/13	mar 12/03/13
Especificación casos	5 horas	mié 13/03/13	jue 14/03/13
Diagrama de clases	3 horas	jue 14/03/13	jue 14/03/13
Diseño ciclo 2	2,5 días	vie 15/03/13	mar 19/03/13
Diseño de la BD	3 horas	vie 15/03/13	vie 15/03/13
Diseño de las clases	3 horas	vie 15/03/13	lun 18/03/13
Diseño de interfaces	4 horas	lun 18/03/13	mar 19/03/13
Construcción ciclo 2	12,5 días	mar 19/03/13	mar 09/04/13
Implementación	40 horas	mar 19/03/13	jue 04/04/13
Pruebas	7 horas	vie 05/04/13	lun 08/04/13
Despliegue	3 horas	lun 08/04/13	mar 09/04/13
Ciclo 3	17,5 días	mié 10/04/13	vie 03/05/13
Análisis ciclo 3	3 días	mié 10/04/13	vie 12/04/13
Identificación casos	4 horas	mié 10/04/13	mié 10/04/13
Especificación casos	5 horas	jue 11/04/13	vie 12/04/13
Diagrama de clases	3 horas	vie 12/04/13	vie 12/04/13
Diseño ciclo 3	2,5 días	lun 15/04/13	mié 17/04/13
Diseño de la BD	3 horas	lun 15/04/13	lun 15/04/13
Diseño de las clases	3 horas	lun 15/04/13	mar 16/04/13
Diseño de interfaces	4 horas	mar 16/04/13	mié 17/04/13
Construcción ciclo 3	12,5 días	mié 17/04/13	vie 03/05/13
Implementación	40 horas	mié 17/04/13	mar 30/04/13
Pruebas	7 horas	mié 01/05/13	jue 02/05/13
Despliegue	3 horas	jue 02/05/13	vie 03/05/13
Memoria	15 horas	lun 06/05/13	jue 09/05/13
Manuales	15 horas	jue 09/05/13	mié 15/05/13
Defensa	15 horas	mié 15/05/13	mar 21/05/13

3.14 Riesgos

En este apartado se especifican los contratiempos que pueden surgir durante la realización del Trabajo Fin de Grado.

1. **Cambios en los requisitos:** Pueden surgir nuevas necesidades respecto a la funcionalidad inicial que afectaría a la planificación inicial. Como consecuencia habrá que incorporar estos nuevos requisitos al proyecto.
2. **Errores en el diseño:** Es posible que el diseño creado no sea acorde con las necesidades que tiene que cumplir el sistema. Habrá que solucionar los errores y documentarlos.

3. **Errores en la planificación:** Problemas relacionados con una mala previsión de los tiempos del Trabajo Fin de Grado, de forma que se incumplan las fechas de entrega. Si necesitamos un mayor plazo para realizar alguna tarea, se retrasarán los plazos de entrega y afectará negativamente.
4. **Pérdida de información o de archivos:** El contenido de los archivos puede ser corrompido, borrado o perdido por lo tanto, es necesario tomar medidas como por ejemplo, realizar copias de seguridad de forma que se puedan recuperar los archivos de forma rápida.
5. **Falta de recursos:** Se necesitan ciertos permisos de usuario para acceder a la base de datos de la Universidad, para pasar los cortafuegos… Esto provoca retrasos en la realización de las tareas, por lo que habrá que adaptar los tiempos a estos imprevistos.
6. **Baja del proyectante:** Representa aquellos problemas derivados de la ausencia del proyectante por enfermedad u otros compromisos importantes. Si esto sucede, habrá que ajustar las horas de trabajo para mantener los plazos de entrega.
7. **Baja del director del proyecto:** Representa aquellos problemas derivados de la ausencia del director del proyecto. Si esto sucede, habrá que valorar si la ausencia es por un periodo largo y se podría proponer un director alternativo.

4. FASE 1

4.1 Análisis

4.1.1 Definición de la fase 1.

Al finalizar esta fase, la aplicación será capaz de crear categorías, crear servicios y mostrar un esquema con todas las categorías y otro con todos los servicios.

4.1.2 Actores del sistema.

En esta parte de la aplicación solo vamos a tener el rol administrador que será el encargado de crear nuevas categorías y nuevos servicios en la aplicación.

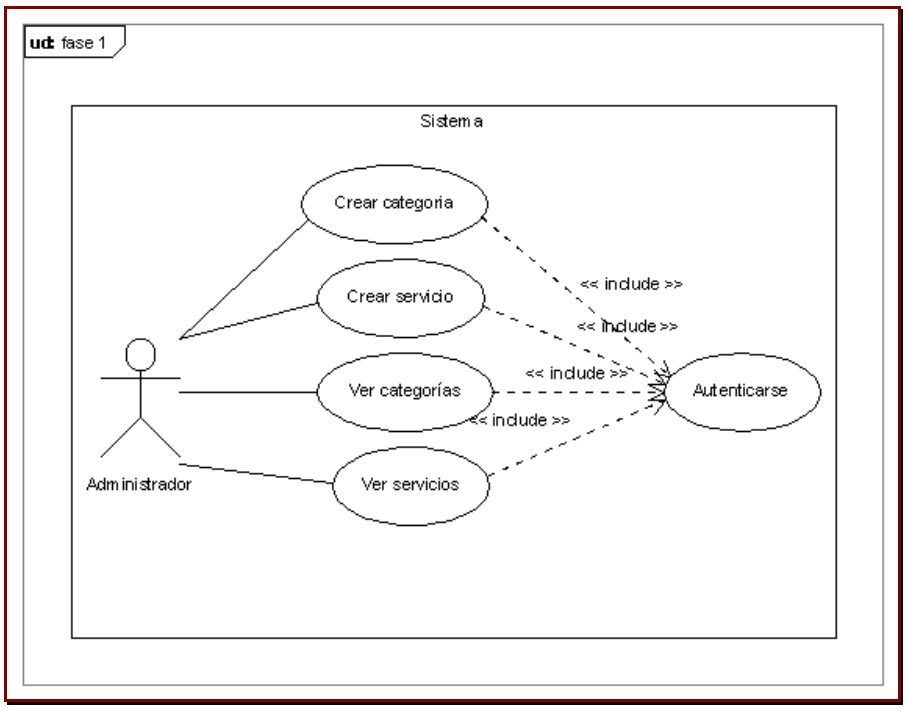
4.1.3 Análisis de requisitos.

En la primera fase de la aplicación, la funcionalidad que se va a desarrollar es la siguiente:

- Creación de nuevas categorías: la aplicación debe permitir al administrador la creación de nuevas categorías que se almacenarán en la base de datos.
- Creación de nuevos servicios: la aplicación debe permitir al administrador la creación de nuevos servicios que se almacenarán en la base de datos.
- Ver categorías: el sistema debe permitir al administrador observar las categorías que forman parte de la aplicación y las relaciones entre sí.
- Ver servicios: el sistema debe permitir al administrador observar los servicios creados y todos los atributos de cada servicio.

4.1.4 Identificación de los casos de uso.

Del primer ciclo de la aplicación podemos obtener el siguiente diagrama de casos de uso:



1. Crear categoría: El administrador podrá crear nuevas categorías introduciendo un identificador, una descripción y si la tuviera, la categoría padre.
2. Crear servicio: El administrador podrá crear nuevos servicios introduciendo un identificador, una descripción, un encargado, una nota, un contacto, un fichero adjunto, un enlace y la categoría padre.
3. Ver categorías: El administrador podrá obtener un esquema de las categorías existentes.
4. Ver servicios: El administrador podrá observar, en una tabla, todos los servicios que existen con sus atributos.
5. Autenticarse: Para poder llevar a cabo estas acciones el administrador deberá estar autenticado y tener permisos aunque la autenticación de usuarios no es algo que se vaya a desarrollar en el proyecto ya que la restricción de acceso la gestiona el repositorio de aplicaciones del servicio informático.

4.1.5 Especificación de los casos de uso.

Crear categoría:

Descripción:	El usuario podrá crear nuevas categorías
Actores:	Administrador del sistema
Precondición:	El usuario debe estar autenticado en el sistema y tener permisos de administrador.
Flujo normal:	1. El administrador accede al portal y se autentica. 2. Dentro de la sección categorías, hace clic en nueva categoría e introduce los datos que le solicita el formulario. 3. El sistema almacena los datos en la tabla adecuada.
Postcondición:	Se añade una nueva categoría a la aplicación.

Crear servicio:

Descripción:	El usuario podrá crear nuevos servicios
Actores:	Administrador del sistema
Precondición:	El usuario debe estar autenticado en el sistema y tener permisos de administrador.
Flujo normal:	1. El administrador accede al portal y se autentica. 2. Dentro de la sección servicios, hace clic en nuevo servicio e introduce los datos que le solicita el formulario.

	3. El sistema almacena los datos en la tabla adecuada.
Postcondición:	Se añade un nuevo servicio a la aplicación.

Ver categorías:

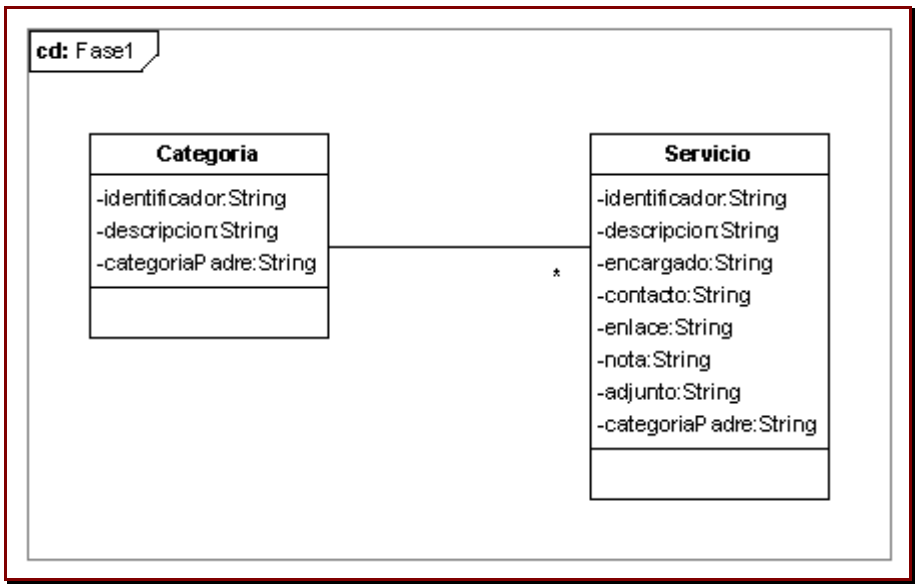
Descripción:	El usuario podrá ver un esquema con las categorías existentes
Actores:	Administrador del sistema
Precondición:	El usuario debe estar autenticado en el sistema y tener permisos de administrador.
Flujo normal:	1. El administrador accede al portal y se autentica. 2. Dentro de la sección categorías, hace clic ver. 3. El sistema muestra un esquema con todas las categorías.

Ver servicios:

Descripción:	El usuario podrá ver en una tabla todos los servicios existentes con los atributos.
Actores:	Administrador del sistema
Precondición:	El usuario debe estar autenticado en el sistema y tener permisos de administrador.
Flujo normal:	1. El administrador accede al portal y se autentica. 2. Dentro de la sección servicios, hace clic ver. 3. El sistema muestra una tabla con los servicios.

4.1.6 Diagrama de clases.

A continuación se muestran las clases del modelo conceptual:



- Clase categoría: representa una categoría. Los atributos de esta clase son el identificador de la categoría, la descripción de la categoría, y el identificador de la categoría padre si es que la tiene. Una categoría padre es aquella que no pertenece a su vez a otra categoría.

- Servicio: representa un servicio. Los atributos de la clase servicio son el identificador del servicio, la descripción y los datos del servicio, encargado, contacto, enlace, nota, archivos adjuntos si los hubiera y la categoría padre.

Una categoría puede tener cero o varios servicios y un servicio pertenece siempre a una categoría.

4.2 Diseño

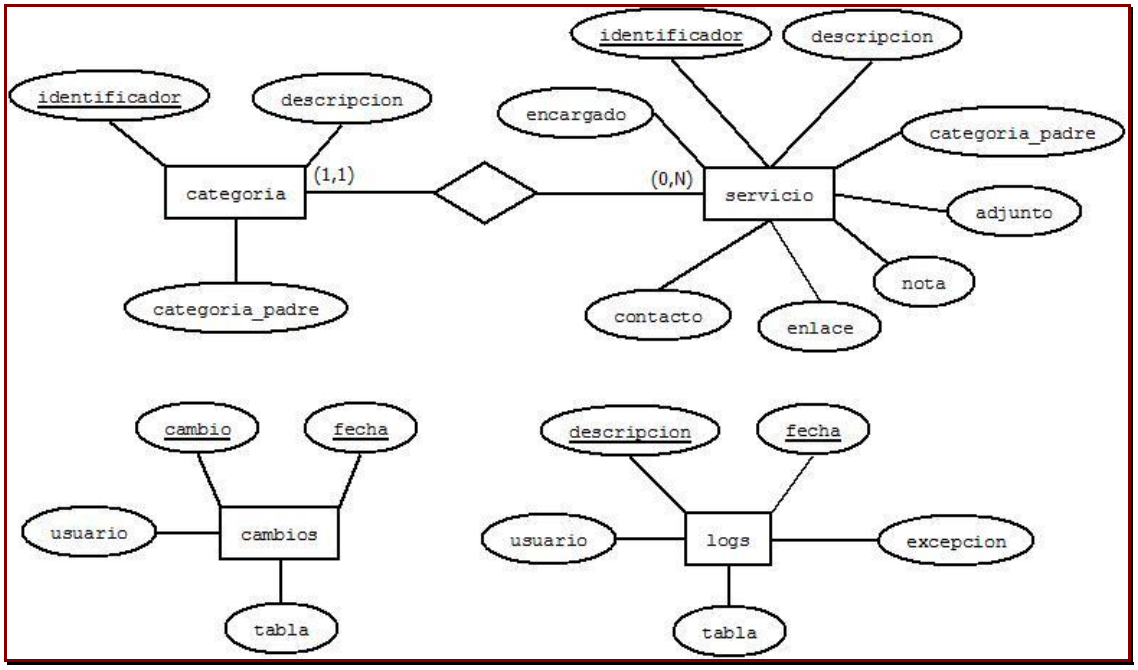
4.2.1 Arquitectura del sistema.

La arquitectura que se utilizará para la aplicación será la arquitectura de tres capas. Se ha elegido esta arquitectura de desarrollo por la ventaja que ofrece el desarrollar una aplicación en varios niveles, distribuyendo el trabajo en las diferentes capas y en caso de que se requiriera algún cambio, se puede ejecutar fácilmente identificando la capa donde debe efectuarse.

Para la capa de presentación se va a utilizar JSP, para la capa de negocio se utilizará Java y para la capa de persistencia se utilizará Java que conectará una Base de Datos Oracle.

4.2.2 Diseño de la Base de datos.

Diagrama de entidad/relación



Para la aplicación se ha creado un esquema de base de datos llamado catálogo que contiene las siguientes tablas:

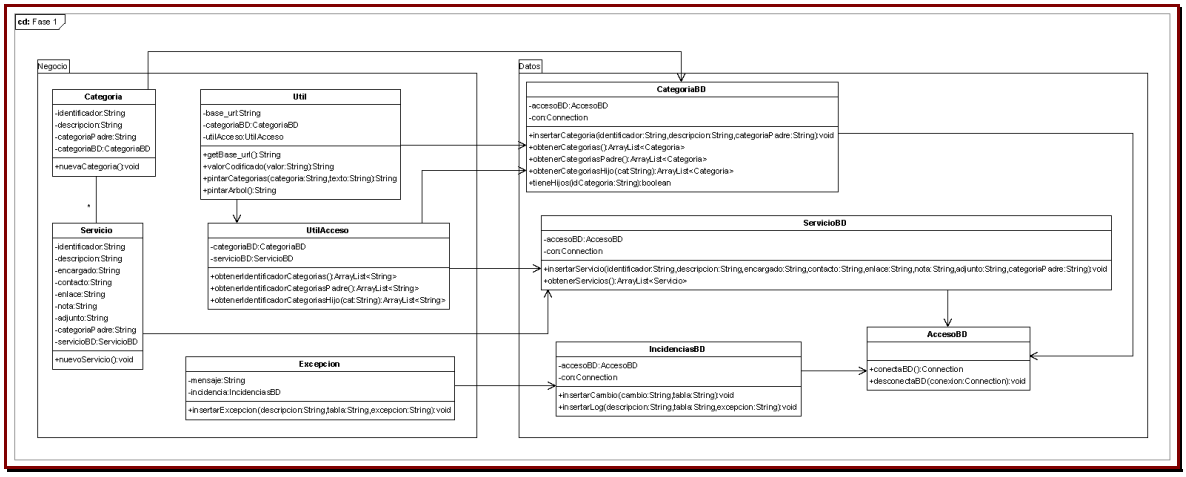
- Categoría: contiene toda la información necesaria para las categorías.
- Servicio: contiene todos los datos de los servicios.
- Cambios: en esta tabla se reflejan los cambios que se hacen en la aplicación, es decir, cada vez que se crea una nueva categoría o servicio se deja constancia en esta tabla.
- Logs: cada vez que la aplicación lanza una excepción, se guarda el registro en esta tabla.

Existe una relación entre las entidades categoría y servicio ya que una categoría puede tener varios o no tener servicios y un servicio pertenece siempre a una categoría únicamente.

Normalización

La base de datos está en forma normal de Boyce-Codd ya que todos sus atributos son monovaluados, los atributos que no forman parte de ninguna clave dependen de forma completa de la clave principal, no existe ninguna dependencia funcional transitiva entre los atributos que no son clave y cada dependencia funcional no trivial tiene una clave candidata como determinante. Al tratarse de una base de datos tan pequeña, es normal que esté normalizada.

4.2.3 Diseño de las clases.



Para la primera fase de la aplicación, las clases utilizadas son las que se han mostrado en el diagrama de clases anterior. Estas clases están divididas en dos paquetes, uno denominado datos y otro llamado negocio. En el diagrama anterior no se han añadido los métodos get y set por motivos de claridad.

En el paquete datos se encuentran las clases que conectan con la base de datos. Estas clases son:

- AccesoBD: la clase que actúa directamente con la base de datos. Contiene los métodos
    - **conectaBD** que conecta con la base de datos.
    - **desconectaBD** que se desconecta de la base de datos.
- CategoriaBD: en esta clase se encuentran los métodos que actúan con la tabla categoría. Los métodos de esta clase son
    - **insertarCategoria** que almacena una nueva categoría en la base de datos.
    - **obtenerCategorias** devuelve un vector con todas las categorías.
    - **obtenerCategoriasPadre** que al igual que el método anterior, devuelve un vector con categorías pero en este caso con las categorías padre únicamente.
    - **obtenerCategoriasHijo** que devuelve un vector con todas las categorías cuya categoría padre sea la que se introduce por parámetro.
    - **tieneHijos** que es una función booleana que devuelve verdadero o falso dependiendo de si el valor introducido como parámetro tiene hijos o no.
- ServicioBD: en esta clase se encuentran los métodos que actúan con la tabla servicio. Los métodos de esta clase son
    - **insertarServicio** que almacena un nuevo servicio en la tabla correspondiente.
    - **obtenerServicios** que devuelve un vector con todos los servicios almacenados.

- IncidenciasBD: esta clase contiene los métodos necesarios para registrar tanto los cambios efectuados en la aplicación como los errores que devuelva la misma. Los métodos de la clase son

    - **insertarCambio** que cada vez que se crea una nueva categoría o servicio almacena en la tabla cambios un registro de la acción que se ha llevado a cabo.

    - **insertarLog**, cada vez que la aplicación genera una excepción se registra en la tabla logs.

Por otra parte se encuentra el paquete negocio con las siguientes clases:

- Categoría: esta clase, aparte de los métodos para acceder y modificar los atributos de la clase, contiene métodos para acceder desde la parte de presentación a la parte de datos. Entre los métodos de esta clase se encuentran:

    - **nuevaCategoria**: que recoge los parámetros de la página jsp y llama al método insertarCategoria.

- Servicio: al igual que la clase anterior, servicio contiene los métodos para acceder y modificar los atributos de la clase servicio, pero además contiene métodos para acceder desde la capa de presentación a la de datos. Los métodos de esta clase son:

    - **nuevoServicio**: recoge los datos de servicio a través de la jsp correspondiente y llama al método insertarServicio.

    - **obtenerServicios**: devuelve un vector con todos los servicios.

- Util: se trata de una clase con métodos auxiliares. Entre los métodos de esta clase están:

    - **getBase_url**: que devuelve la dirección absoluta del proyecto. De esta forma, cuando subamos el proyecto al servidor no habrá que cambiar en todas las páginas la dirección base del proyecto, con cambiarla una vez funcionará perfectamente.

    - **valorCodificado**: se le pasa un valor recogido de una página jsp y devuelve el mismo valor codificado correctamente.

    - **pintarCategorias** y **pintarArbol**: ambos métodos son para pintar la lista de categorías.

- UtilAcceso: se trata también de una clase auxiliar pero en este caso, contiene métodos para acceder desde la capa de presentación a la de datos. Métodos que no pueden formar parte de ninguna otra clase. Los métodos de esta clase son:

    - **obtenerIdentificadorCategorias, obtenerIdentificadorCategoriasPadre** y **obtenerIdentificadorCategoriasHijo**: los tres métodos devuelven un vector de

String con todos los identificadores de las categorías, las categorías padre y las categorías hijo respectivamente.

- Excepcion: se trata de la clase que se encarga de mostrar el mensaje correcto cada vez que se genera una excepción. Entre los métodos de esta clase se encuentran:
    - **getMensaje** y **setMensaje**: que devuelve y cambia el mensaje de la excepción respectivamente.
    - **insertarExcepcion**: que recoge los datos generados por la excepción y llama al método insertarLog.

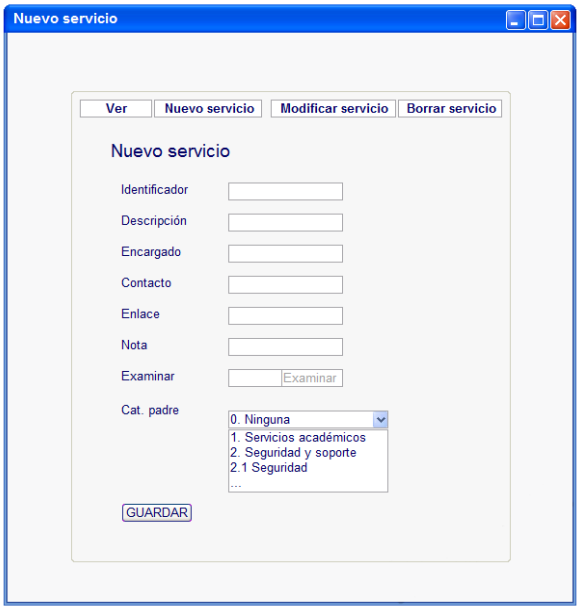
4.2.4 Diseño de interfaces.

En este apartado se mostrarán los prototipos utilizados para la implementación de las interfaces finales. Para elaborar los prototipos de interfaz, se ha utilizado el programa Mockup Screens. Para elegir el diseño de las interfaces se ha tomado como referencia el resto de aplicaciones del servicio informático ya que va a ser una aplicación cuyos administradores van a ser los usuarios del servicio informático.

Índice del administrador

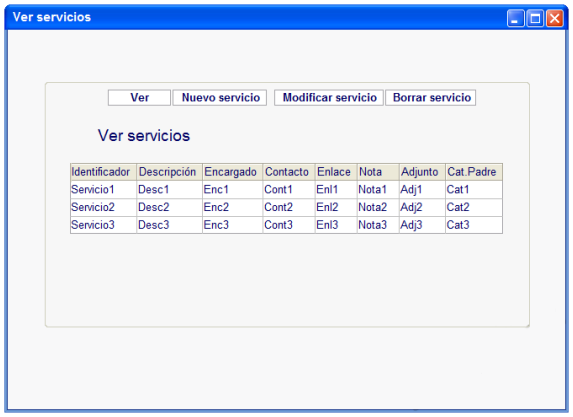
Nueva categoría

Nuevo servicio



Ver categorías

Ver servicios


Ver servicios
Ver
Nuevo servicio
Modificar servicio
Borrar servicio
Ver servicios
Identificador Descripción Encargado Contacto Enlace Nota Adjunto Cat.Padre
Servicio1 Desc1 Enc1 Cont1 Enl1 Nota1 Adj1 Cat1
Servicio2 Desc2 Enc2 Cont2 Enl2 Nota2 Adj2 Cat2
Servicio3 Desc3 Enc3 Cont3 Enl3 Nota3 Adj3 Cat3

4.3 Construcción

4.3.1 Implementación

Tecnologías empleadas

Durante el desarrollo de la aplicación se han ido utilizando distintas tecnologías para suplir las necesidades que se daban en la realización de las distintas partes de este. Estas tecnologías son:

JSP y Java:

Para programar la aplicación se ha utilizado Java para la parte de programación y JSP para la parte visual. Se han elegido estas tecnologías porque la mayoría de las aplicaciones del Servicio Informático son Java y por lo tanto es más sencilla la integración de las distintas aplicaciones si

están en el mismo lenguaje de programación. Además Java es un lenguaje muy seguro y robusto.

CSS

Para los estilos de la parte visual de la aplicación se han creado hojas de estilo CSS, de esta forma se puede utilizar la misma hoja de estilos en diferentes páginas sin necesidad de repetir código.

Oracle

Para la gestión de bases de datos se ha utilizado Oracle ya que en el Servicio Informático es el sistema gestor de bases de datos que se utiliza y como la aplicación accede a algunas de las tablas ya creadas anteriormente, es necesario utilizar Oracle.

Herramientas de desarrollo

Netbeans IDE 7.3

Para la programación de la aplicación he utilizado el IDE de desarrollo Netbeans 7.3. He elegido este entorno de desarrollo porque es un producto libre, gratuito y sin restricciones de uso, es multiplataforma y contiene una gran comunidad de usuarios.

Apache Tomcat 7.0.39

Para las pruebas locales se ha utilizado el servidor Apache Tomcat ya que es el servidor que se utilizará cuando se despliegue la aplicación en producción.

Construcción de la base de datos

Como se ha indicado en la parte de análisis, se ha creado el esquema Catálogo que contiene las tablas categoría, servicio, cambios y logs. He utilizado el sistema de gestión de bases de datos Oracle ya que es el utilizado por la Universidad. Se utiliza Oracle por la estabilidad y porque tiene soporte multiplataforma.

En la Universidad hay varios servidores de bases de datos dedicados a diferentes funciones. En el caso de esta aplicación, de momento se ha utilizado el servidor de desarrollo, es decir, el que se utiliza para hacer pruebas. De esta forma si tuviéramos algún problema con la base de datos, no afectaría al resto de programas de la Universidad.

Para desarrollar la aplicación hemos creado un usuario, “Catalogo”, con los permisos necesarios para gestionar las tablas que tenga su esquema.

Lógica de la aplicación

A continuación voy a explicar las partes más destacables del código.

Librería ur.jar.

El Servicio Informático de la Universidad tiene una librería que contiene los métodos necesarios para interactuar con todos los repositorios que tienen.

En concreto, para esta fase del proyecto utilizo la parte de conexión a las bases de datos. Para la conexión a la base de datos se necesita crear un fichero properties con la url de la base de datos, el número máximo de conexiones, el fichero de logs y algunos datos más como el usuario y la contraseña:

```
drivers=oracle.jdbc.driver.OracleDriver
#logfile=/tmp/log.txt
logfile=/catalogo.log

# Desarrollo
bd.url=url:desarrollo

#Produccion
#bd.url=url:produccion

bd.maxconn=20
bd.initconn=1
bd.timeout=2
bd.user=usuario
bd.password=contraseña

#(1=hacer debug, 0=no hacer debug)
bd.debug=0
bd.database=oracle
#0=rapido (sin soporte firewall), 1=lento (con soporte firewall)
bd.ckmethod=1
```

De esta forma, para conectarnos con la base de datos únicamente tenemos que llamar a un método de la librería ur.jar:

```
GruposConexionesBD grupoCon = GruposConexionesBD.obtenerInstancia();
con = grupoCon.obtenerConexion("bd");
```

Paginación de tablas:

Para mostrar los servicios se utilizan tablas. Para controlar el número de servicios que se muestran por cada página se ha utilizado una librería jsp que facilita la tarea de paginar los datos obtenidos. Esta librería es DisplayTag y su uso es muy sencillo. Basta con añadir dos ficheros JavaScript y luego se añade el siguiente script:

```
$(document).ready(function() {
    $('#example').dataTable();
});
```

Para que la tabla se muestre paginada, a la hora de crearla habrá que ponerle el mismo id que se ha puesto en el script superior:

```
<table cellpadding="0" cellspacing="0" border="0" class="display" id="example">
```

Seguridad de la aplicación

Para controlar el acceso a la aplicación, se va a limitar el acceso permitiendo solo el uso de esta parte de la aplicación a ciertos usuarios del Servicio Informático. Al acceder a la aplicación los usuarios se autenticarán con la CUASI y si tienen permiso para utilizarla, podrán acceder.

Interfaces definitivas del sistema

A continuación se mostrarán algunas de las interfaces creadas para esta fase de la aplicación. Son prácticamente iguales a los prototipos creados en la fase de análisis porque se enseñaron los prototipos al personal que va a utilizar la aplicación y éstos dieron el visto bueno.

Interfaz índice administrador

Interfaz de ver categorías

Interfaz de ver servicios

Interfaz de crear categorías

Interfaz de crear servicios

4.3.2 Pruebas

En este apartado se realizarán las pruebas para testar diferentes aspectos de la aplicación. Se han realizado las pruebas unitarias y de integración de manera conjunta para facilitar la tarea. De esta manera probamos conjuntamente las tres capas de la aplicación.

Para las pruebas de la capa de presentación de la aplicación, se ha probado en diferentes navegadores como Internet Explorer 9, Mozilla Firefox 21, Opera y Chrome.

Para probar que todos los métodos funcionan correctamente, he creado una clase de pruebas que llama a los métodos principales.

```
public class prueba {
    public static void main(String[] args) {
        try {
            CategoriaBD categoriaBD = new CategoriaBD();
            ServicioBD servicioBD = new ServicioBD();
            categoriaBD.insertarCategoria("categoriaPrueba", "Descripcion", null);
            servicioBD.insertarServicio("servicioPrueba", "Descripcion", "aa", "bb", "cc",
                    "dd", "ee", "categoriaPrueba", 1);
            System.out.println("¿Tiene hijos categoriaPrueba? "+categoriaBD.tieneHijos("categoriaPrueba"));
            ArrayList<Categoria> cat = categoriaBD.obtenerCategorias();
            ArrayList<Servicio> serv = servicioBD.obtenerServicios();
            for(int i=0;i<cat.size();i++){
                System.out.println("Categoría: "+cat.get(i).getIdentificador());
            }
            for(int j=0;j<serv.size();j++){
                System.out.println("Servicio: "+serv.get(j).getIdentificador());
            }

        } catch (SQLException ex) {
            Logger.getLogger(prueba.class.getName()).log(Level.SEVERE, null, ex);
        }

    }
}
```

En esta clase de pruebas se han creado una nueva categoría, un nuevo servicio, se muestran todas las categorías y los servicios y se pregunta a una categoría si tiene hijos. El resultado de la ejecución de esta clase es el siguiente:

```
arch Results | Output - Catalogo (run)
run:
LA CATEGORÍA categoriaPrueba SE HA CREADO CORRECTAMENTE
EL SERVICIO servicioPrueba SE HA CREADO CORRECTAMENTE
¿Tiene hijos categoriaPrueba? false
conexion com.mysql.jdbc.Connection@50ef5502
Categoría: Cat1
Categoría: Cat1.1
Categoría: cat1.2
Categoría: Cat1.2.1
Categoría: Cat2
Categoría: Cat2.1
Categoría: Cat2.2
Categoría: Cat3
Categoría: categoriaPrueba
Categoría: catPrueba
Servicio: serPrueba
Servicio: Servicio1
Servicio: Servicio2
Servicio: servicioPrueba
BUILD SUCCESSFUL (total time: 0 seconds)
```

Todos los métodos han funcionado correctamente.

El resto de métodos de la aplicación se han probado haciendo pruebas directamente utilizando la interfaz de la aplicación.

4.3.3 Conclusiones

Tras finalizar la primera fase, la aplicación se ha entregado al cliente que en todo momento ha supervisado el desarrollo de la misma. Esta fase ha cumplido las expectativas por lo tanto no hay que añadir ninguna modificación.

5. FASE 2

5.1 Análisis

5.1.1 Definición de la fase 2.

Al finalizar esta fase, la aplicación nos permitirá modificar y borrar categorías y servicios y permitirá a los usuarios ver el catálogo de servicios de la aplicación.

5.1.2 Actores del sistema.

En esta parte de la aplicación, además del usuario administrador tendremos al usuario visitante.

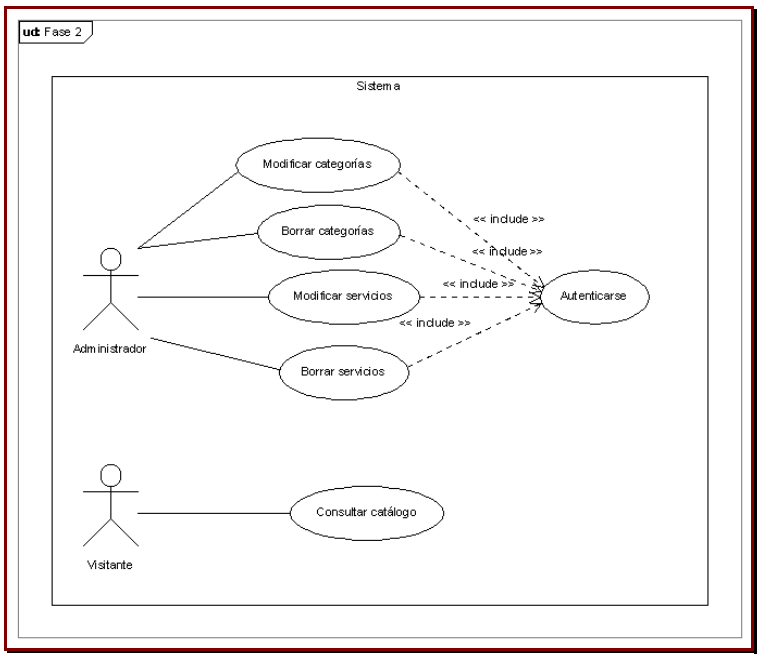
5.1.3 Análisis de requisitos.

En la segunda fase de la aplicación, la funcionalidad que se va a desarrollar es la siguiente:

- Modificar categorías: la aplicación debe permitir que se modifiquen las categorías que han sido creadas anteriormente.
- Borrar categorías: el sistema debe permitir que el administrador elimine las categorías creadas con anterioridad.
- Modificar servicios: la aplicación debe permitir que el administrador modifique los servicios.
- Borrar servicios: el sistema debe permitir al administrador borrar los servicios creados anteriormente.
- Consultar catálogo: la aplicación debe permitir a los usuarios visitantes de la web consultar el catálogo de servicios.

5.1.4 Identificación de los casos de uso.

De la segunda fase de la aplicación obtenemos el siguiente diagrama de casos de uso:



Casos de uso del administrador:

1. Modificar categoría: El administrador podrá modificar las categorías creadas cambiando el identificador, la descripción o la categoría padre.
2. Modificar servicio: El administrador podrá modificar los datos de los servicios ya creados.
3. Borrar categorías: El administrador podrá borrar cualquier categoría creada.
4. Borrar servicios: El administrador podrá borrar cualquiera de los servicios creados.

Casos de uso del usuario visitante:

1. Consultar catálogo: los visitantes podrán consultar el catálogo de servicios.

5.1.5 Especificación de los casos de uso.

Modificar categoría:

Descripción:	El administrador podrá modificar las categorías
Actores:	Administrador del sistema
Precondición:	El usuario debe estar autenticado en el sistema y tener permisos de administrador.
Flujo normal:	1. El administrador accede al portal y se autentica. 2. Dentro de la sección categorías, hace clic en modificar y cambia los datos que desee. 3. El sistema almacena los cambios en la base de datos.
Postcondición:	Se modifica la categoría seleccionada.

Borrar categoría:

Descripción:	El administrador podrá borrar categorías
Actores:	Administrador del sistema
Precondición:	El usuario debe estar autenticado en el sistema y tener permisos de administrador.
Flujo normal:	1. El administrador accede al portal y se autentica. 2. Dentro de la sección categorías, hace clic en borrar y selecciona la categoría que desea borrar. 3. El sistema elimina la categoría elegida de la base de datos.
Postcondición:	Se elimina la categoría.

Modificar servicio:

Descripción:	El administrador podrá modificar los servicios
Actores:	Administrador del sistema
Precondición:	El usuario debe estar autenticado en el sistema y tener permisos de administrador.
Flujo normal:	1. El administrador accede al portal y se autentica. 2. Dentro de la sección servicios, hace clic en modificar y cambia los datos que desee. 3. El sistema almacena los cambios en la base de datos.
Postcondición:	Se modifica el servicio seleccionada.

Borrar servicio:

Descripción:	El administrador podrá borrar servicios
Actores:	Administrador del sistema
Precondición:	El usuario debe estar autenticado en el sistema y tener permisos de administrador.
Flujo normal:	1. El administrador accede al portal y se autentica. 2. Dentro de la sección servicios, hace clic en borrar y selecciona el servicio que desea borrar. 3. El sistema elimina el servicio elegido de la base de datos.
Postcondición:	Se elimina el servicio.

Consultar catálogo:

Descripción:	Los visitantes podrán consultar el catálogo de servicios
Actores:	Cualquier usuario que acceda a la página Web de la Universidad de La Rioja
Flujo normal:	1. El usuario accede a la URL de la página Web y podrá consultar todos los servicios disponibles.

5.1.6 Diagrama de clases.

El diagrama de clases del modelo conceptual es el mismo que el de la fase 1.

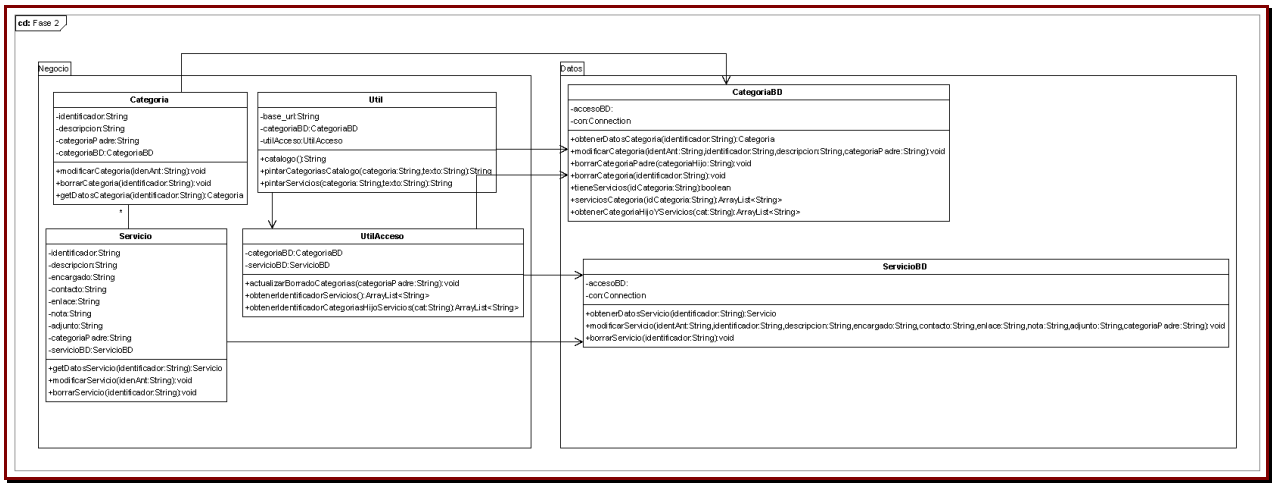
5.2 Diseño

5.2.1 Diseño de la Base de datos.

Para esta fase se han utilizado las mismas tablas que en la fase 1.

5.2.2 Diseño de las clases.

Para la segunda fase de la aplicación se han utilizado las mismas clases que en la primera fase pero añadiendo nuevos métodos. El diagrama de clases de esta fase es el siguiente:



En el diagrama solo aparecen los nuevos métodos por claridad. Estos métodos son:

- CategoriaBD:
    - **obtenerDatosCategoria** devuelve los datos de una categoría concreta.
    - **modificarCategoria** modifica los datos de la categoría.
    - **borrarCategoriaPadre** pone a null la categoría padre de la categoría que se ha introducido como parámetro.
    - **borrarCategoria** elimina la categoría seleccionada.
    - **tieneServicios** devuelve verdadero o falso dependiendo de si la categoría tiene servicios.
    - **serviciosCategoria** devuelve un vector con todos los servicios de una categoría.
    - **obtenerCategoriasHijoYServicios** devuelve un vector con todas las categorías hijo y los servicios de una categoría.
- ServicioBD:
    - **obtenerDatosServicio** devuelve todos los datos de un servicio concreto.

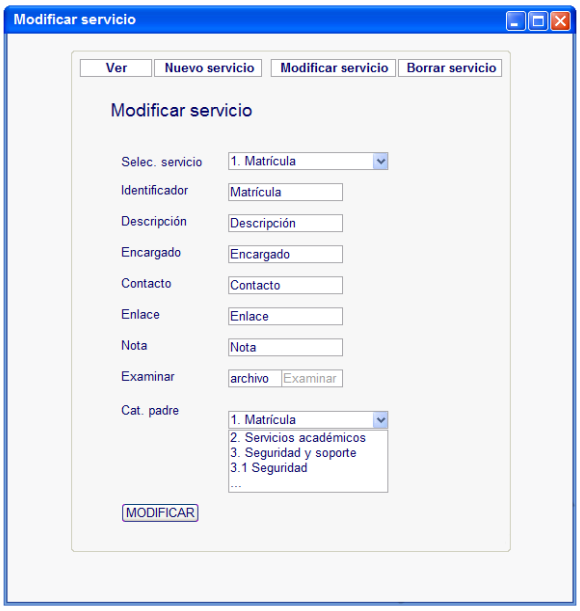
- **modificarServicio** modifica los datos de un servicio.
- **borrarServicio** elimina un servicio seleccionado de la base de datos.

Los nuevos métodos de las clases del paquete negocio son:

- Categoría:
  - **modificarCategoria**: recoge los valores de la página jsp y llama al método de categoriaBD.
  - **borrarCategoria** recoge de la página jsp el identificador de la categoría que se desea borrar y llama al método de borrar categoría de la clase categoriaBD.
  - **getDatosCategoria** se recoge el identificador de la categoría y se llama al método de la clase categoriaBD para obtener los datos de la categoría.
- Servicio:
  - **getDatosServicio**: se recoge el identificador del servicio y se llama al método de la clase servicioBD obtenerDatosServicio.
  - **modificarServicio**: se recogen los datos del servicio de la página jsp y se llama al método modificarServicio de servicioBD.
  - **borrarServicio**: se recoge el identificador del servicio que se desea borrar y se llama al método borrarServicio de servicioBD.
- Util:
  - **catalogo, pintarCategoriasCatalogo y pintarServicios:** los tres métodos son para recuperar las categorías y los servicios y pintar por pantalla el catálogo de servicios.
- UtilAcceso:
  - **actualizarBorradoCategorias:** recoge en un vector todas las categorías hijo de una categoría y para cada categoría hijo se pone a null el campo categoría_padre.
  - **obtenerIdentificadorServicios:** devuelve un vector con todos los identificadores de los servicios.
  - **obtenerIdentificadorCategoriasHijoServicios:** recoge el identificador de la categoría y llama al método obtenerCategoriasHijoYServicios de categoriaBD.

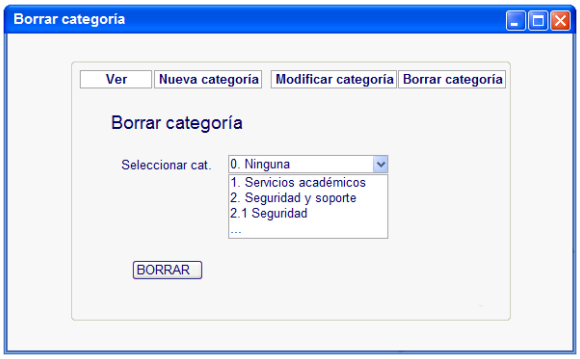
5.2.3 Diseño de interfaces.

Modificar servicio

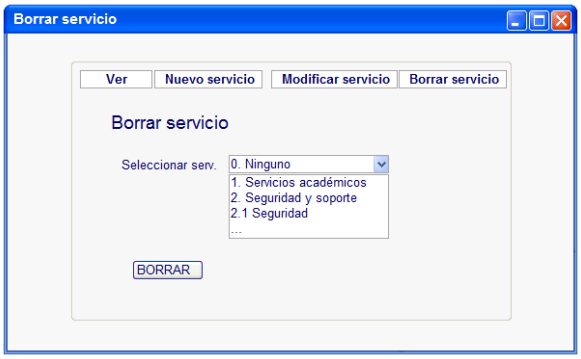


Modificar categoría

Borrar categorías



Borrar servicios



Catálogo de servicios

5.3 Construcción

5.3.1 Implementación

Tecnologías empleadas

Las tecnologías empleadas en esta fase de la aplicación son las mismas que las que se utilizaron para implementar la fase 1.

Construcción de la base de datos

Como se han utilizado las mismas tablas que en la fase 1, no ha sido necesario implementar nada nuevo en la construcción de la base de datos.

Lógica de la aplicación

Hay que destacar la manera en la que se muestra el catálogo de servicios. La única posibilidad para mostrar un árbol en jsp era mediante un applet, pero al no encontrar ninguno que se adecuara a nuestras necesidades decidimos que lo mejor era cargar las categorías y los servicios en forma de una lista ordenada HTML. Estos son los métodos que he creado para pintar por pantalla el catálogo:

```
public String catalogo(){
    String texto = "";
    try {
        ArrayList<String> padres = utilAcceso.obtenerIdentificadorCategoriasPadre();
        texto +="<ul>";
        for (int i = 0; i < padres.size(); i++) {
            texto += "<li>" + padres.get(i);
            if (categoriaBD.tieneHijos(padres.get(i))) {
                texto = this.pintarCategoriasCatalogo(padres.get(i), texto);
            }
            else if(categoriaBD.tieneServicios(padres.get(i))){
                texto = this.pintarServicios(padres.get(i), texto);
            }

            texto += "</li>";
        }
        texto += "</ul>";
        return texto;
    } catch (SQLException ex) {
        Logger.getLogger(Util.class.getName()).log(Level.SEVERE, null, ex);
    }
    finally{
        return texto;
    }
```

Esta es la función principal que a su vez llama a otros dos métodos:

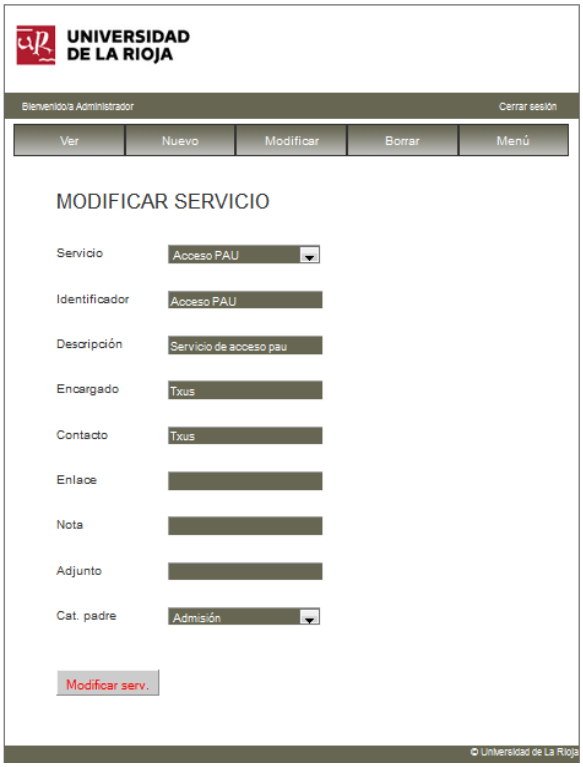
```
public String pintarCategoriasCatalogo(String categoria, String texto) throws SQLException {
    ArrayList<String> catHijos = utilAcceso.obtenerIdentificadorCategoriasHijoServicios(categoria);
    texto += "<ul>";
    for (int i = 0; i < catHijos.size(); i++) {
        //System.out.println("<li>"+catHijos.get(i));
        if (categoriaBD.tieneHijos(catHijos.get(i))) {
            texto += "<li>" + catHijos.get(i);
            texto = pintarCategoriasCatalogo(catHijos.get(i), texto);
        } else if(categoriaBD.tieneServicios(catHijos.get(i))){
            texto = this.pintarServicios(catHijos.get(i), texto);
        }
        else {
            texto += "<li>" + catHijos.get(i);
        }
        texto += "</li>";
    }
    texto += "</ul>";
    return texto;
}
```

```
public String pintarServicios(String idCategoria, String texto){
    try {
        ArrayList<String> servicios = categoriaBD.serviciosCategoria(idCategoria);
        texto += "<ul>";
        for(int i=0; i<servicios.size(); i++){
            texto +="<li>"+servicios.get(i)+"</li>";
        }
        texto +="</ul>";
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
    finally{
        return texto;
    }
}
```

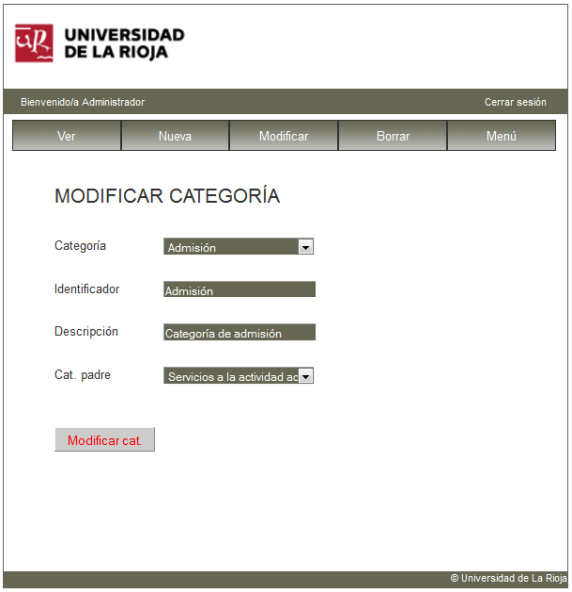
Interfaces definitivas del sistema

A continuación se mostrarán algunas de las interfaces creadas para esta fase de la aplicación. Ha habido pocos cambios respecto a los prototipos ya que el cliente dio el visto bueno al diseño de las interfaces realizado en la fase de análisis.

Interfaz de modificar servicios



Interfaz de modificar categorías



Interfaz de borrar categorías

Interfaz de borrar servicios

Interfaz de mostrar catálogo

5.3.2 Pruebas

Para facilitar la tarea se han vuelto a realizar las pruebas unitarias y de integración conjuntamente.

Como en la anterior fase, la capa de presentación se ha probado en los diferentes navegadores.

De nuevo, se ha creado una clase de prueba para comprobar el funcionamiento correcto de los métodos principales.

```
public class prueba {
    public static void main(String[] args) {
        try {
            CategoriaBD categoriaBD = new CategoriaBD();
            ServicioBD servicioBD = new ServicioBD();

            Categoria c = categoriaBD.obtenerDatosCategoria("categoriaPrueba");
            System.out.println("DATOS DE LA CATEGORÍA: "+c.getIdentificador()+"; "+c.getDescripcion());
            categoriaBD.modificarCategoria("categoriaPrueba", "CategoriaPruebaMod", "Descripcion", null);
            Categoria a = categoriaBD.obtenerDatosCategoria("CategoriaPruebaMod");
            System.out.println("DATOS DE LA CATEGORÍA MODIF: "+a.getIdentificador()+"; "+a.getDescripcion());
            System.out.println("TIENE SERVICIOS? "+categoriaBD.tieneServicios("CategoriaPruebaMod"));
            Servicio s = servicioBD.obtenerDatosServicio("servicioPrueba");
            System.out.println("DATOS DEL SERVICIO: "+s.getIdentificador()+"; "+s.getDescripcion());
            servicioBD.modificarServicio("servicioPrueba", "servicioPruebaMod", "Descripcion", "zz", "yy", "xx",
                    "jj", "tt", "CategoriaPruebaMod");
            Servicio d = servicioBD.obtenerDatosServicio("servicioPruebaMod");
            System.out.println("DATOS DEL SERVICIO MODIF: "+d.getIdentificador()+"; "+d.getDescripcion());

            categoriaBD.borrarCategoria("CategoriaPruebaMod");
            servicioBD.borrarServicio("servicioPruebaMod");
        } catch (SQLException ex) {
            Logger.getLogger(prueba.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
```

En esta clase de pruebas se muestran los datos de una categoría, se modifica esa categoría y se vuelven a mostrar los datos de la nueva categoría, y lo mismo con un servicio. También se borran la categoría y el servicio creados. El resultado de la ejecución de esta clase es el siguiente:

```
run:
DATOS DE LA CATEGORÍA: categoriaPrueba; Descripcion
SE HA MODIFICADO LA CATEGORÍA categoriaPrueba
DATOS DE LA CATEGORÍA MODIF: CategoriaPruebaMod; Descripcion
TIENE SERVICIOS? false
DATOS DEL SERVICIO: servicioPrueba; Descripcion
SE HA MODIFICADO EL SERVICIO servicioPrueba
DATOS DEL SERVICIO MODIF: servicioPruebaMod; Descripcion
SE HA BORRADO LA CATEGORÍA CategoriaPruebaMod
SE HA BORRADO EL SERVICIO servicioPruebaMod
BUILD SUCCESSFUL (total time: 0 seconds)
```

Todos los métodos han funcionado correctamente.

El resto de métodos de la aplicación se han probado haciendo pruebas directamente utilizando la interfaz de la aplicación.

5.3.3 Conclusiones

Tras la finalización de esta fase, el cliente ha recibido la aplicación y ha dado el visto bueno a la parte de la aplicación realizada en esta fase.

6. FASE 3

6.1 Análisis

6.1.1 Definición de la fase 3.

Al finalizar esta fase, además de tener la aplicación completa con los estilos definitivos se podrán consultar los servicios, hacer búsquedas entre los servicios y se controlará el acceso entre distintos tipos de usuario a los distintos servicios.

6.1.2 Actores del sistema.

En esta parte de la aplicación, además del usuario administrador tendremos al usuario visitante.

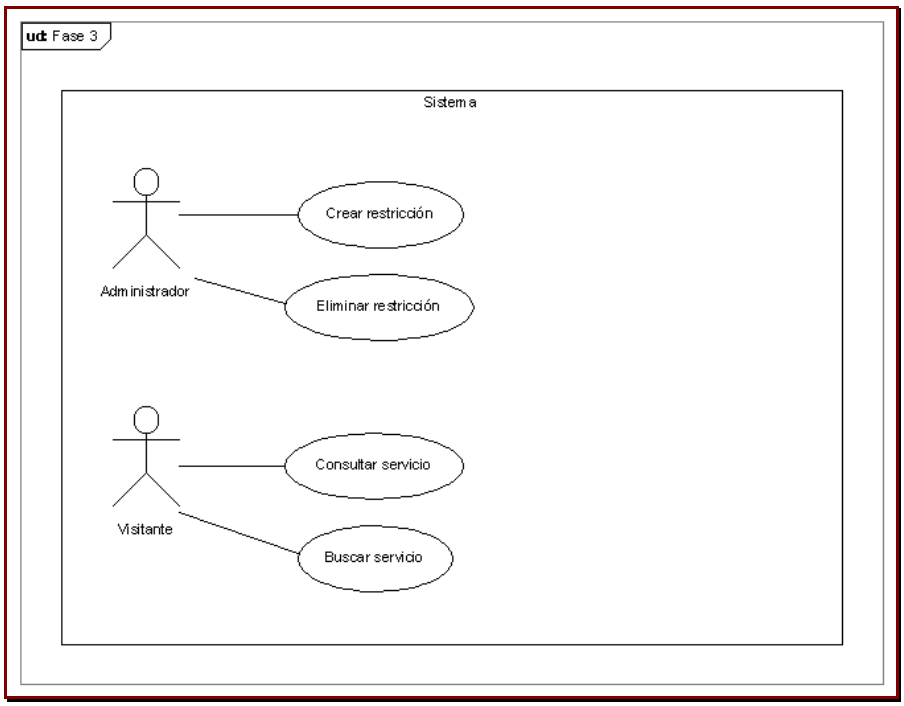
6.1.3 Análisis de requisitos.

En la última fase de la aplicación, la funcionalidad que se va a desarrollar es la siguiente:

- Añadir y borrar acceso a los servicios: el administrador podrá restringir y quitar las restricciones de los distintos servicios.
- Buscar servicios: los usuarios podrán, a través de un buscador, realizar búsquedas entre los servicios.
- Consultar servicios: los visitantes podrán consultar toda la información que tiene un servicio.

6.1.4 Identificación de los casos de uso.

De la tercera fase de la aplicación obtenemos el siguiente diagrama de casos de uso:



Casos de uso del administrador:

1. Crear restricción: El administrador podrá limitar el acceso de los usuarios a los servicios.
2. Eliminar restricción: El administrador podrá eliminar las restricciones de acceso a los servicios.

Casos de uso del usuario visitante:

1. Consultar servicio: los visitantes podrán consultar los datos de los servicios.
2. Buscar servicio: los visitantes podrán realizar búsquedas entre los servicios creados.

6.1.5 Especificación de los casos de uso.

Crear restricción:

Descripción:	El administrador podrá crear nuevas restricciones
Actores:	Administrador del sistema
Precondición:	El usuario debe estar autenticado en el sistema y tener permisos de administrador.
Flujo normal:	1. El administrador accede al portal y se autentica. 2. Dentro de la sección restricciones, hace clic en crear una nueva restricción, selecciona el servicio e introduce los datos necesarios para la restricción. 3. El sistema almacena los cambios en la base de datos.
Postcondición:	El acceso a ese servicio estará restringido para el usuario o grupo de usuarios indicado.

Eliminar restricción:

Descripción:	El administrador podrá eliminar las restricciones creadas con anterioridad.
Actores:	Administrador del sistema
Precondición:	El usuario debe estar autenticado en el sistema y tener permisos de administrador.
Flujo normal:	1. El administrador accede al portal y se autentica. 2. Dentro de la sección restricciones, hace clic en eliminar restricción, selecciona el servicio y selecciona las restricciones que desea borrar.

	3. El sistema elimina la restricción de ese servicio.
Postcondición:	Se elimina la restricción de acceso a ese servicio por el grupo de usuarios seleccionado.

Consultar servicio:

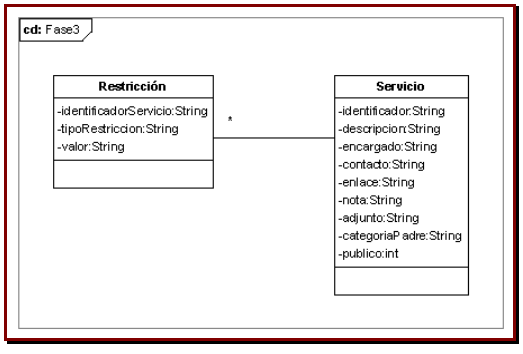
Descripción:	El visitante podrá consultar los datos de los servicios que desee.
Actores:	Actor visitante
Precondición:	
Flujo normal:	1. El visitante accede a la página Web de la aplicación. 2. El visitante selecciona un servicio y hace clic encima del servicio que desee consultar. 3. El sistema muestra al visitante los datos de dicho servicio.

Buscar servicios:

Descripción:	El visitante podrá buscar servicios en la web.
Actores:	Actor visitante
Flujo normal:	1. El visitante accede a la página Web de la aplicación. 2. Accede al buscador e introduce los parámetros de búsqueda que desee. 3. El sistema muestra al visitante los servicios que coincidan con la búsqueda.

6.1.6 Diagrama de clases

Las clases que forman el modelo conceptual son las siguientes:



- Clase restricción: representa cada una de las restricciones que tiene cada servicio. Los atributos de esta clase son el servicio al que se añade esta restricción, el tipo de restricción que se agregará y el valor de la restricción.

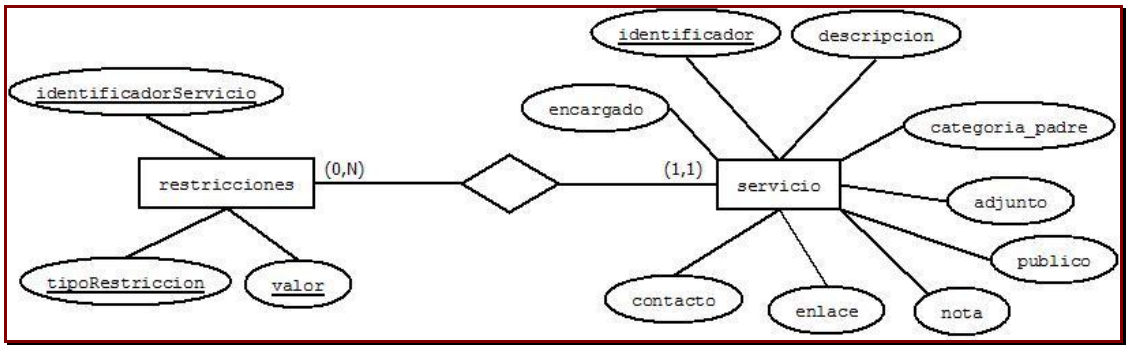
- Servicio: representa a cada uno de los servicios como en la primera fase, pero en esta fase se le ha añadido el atributo público que indica si el servicio tiene restricciones o no.

Un servicio puede tener ninguna o varias restricciones y una restricción pertenece a un servicio.

6.2 Diseño

6.2.1 Diseño de la base de datos.

Diagrama de entidad/relación



En esta fase de la aplicación se ha utilizado la misma base de datos pero se ha añadido una tabla y se ha añadido un nuevo atributo:

- Restricciones: contiene toda la información de cada una de las restricciones que tiene cada servicio.

- Servicio: se ha añadido el atributo público que indica si la tabla es pública o si por el contrario tiene restricciones.

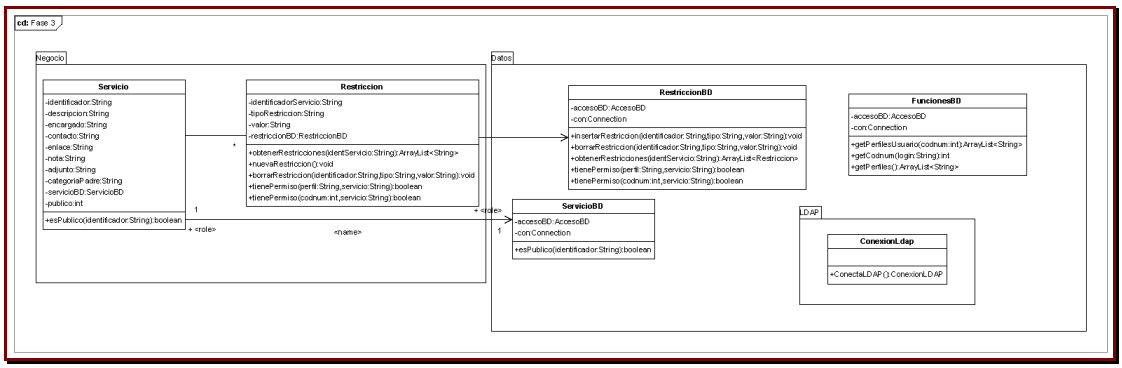
Además de estas tablas ha sido necesario acceder a otras tablas que se encontraban en otra base de datos, pero como esa base de datos no se ha creado para este proyecto porque ya existía, no he agregado esas tablas al diagrama de entidad/relación.

Normalización

Al igual que en la fase 1, la base de datos está en la forma normal de Boyce-Codd.

6.2.2 Diseño de las clases.

El diagrama de clases de la tercera fase de la aplicación es el siguiente:



Solo se han incluido las nuevas clases y los nuevos métodos de las clases antiguas por motivos de claridad.

Las nuevas clases y métodos del paquete datos son:

- RestriccionBD: en esta clase se encuentran todos los métodos que actúan con la tabla restricciones. Estos métodos son:
    - **insertarRestriccion**: almacena una nueva restricción en la base de datos.
    - **borrarRestriccion**: elimina una restricción de acceso de un servicio de la base de datos.
    - **obtenerRestricciones**: devuelve un vector con todas las restricciones de un servicio.
    - **tienePermiso**: devuelve verdadero o falso dependiendo de si un usuario tiene o no permiso de acceso a un servicio.
- FuncionesBD: es una clase con los métodos de acceso al resto de tablas de la base de datos del Servicio Informático. Los métodos de esta clase son:
    - **getPerfilesUsuario**: devuelve un vector con todos los perfiles que tiene un usuario (PSC, PAS…).
    - **getCodnum** : devuelve el codnum de un usuario que es el identificador único del usuario para la base de datos del Servicio Informático.
    - **getPerfiles**: devuelve todos los perfiles existentes en la base de datos del S.I.
- ServicioBD: se ha añadido un nuevo método a esta clase:
    - **esPublico**: que devuelve verdadero o falso dependiendo de si el servicio es público o por el contrario tiene restricciones.

Se ha creado el paquete LDAP dentro de datos ya que por una parte están las clases que acceden a la base de datos y en el paquete LDAP se encuentra la clase para acceder al LDAP. En este paquete se encuentra la siguiente clase:

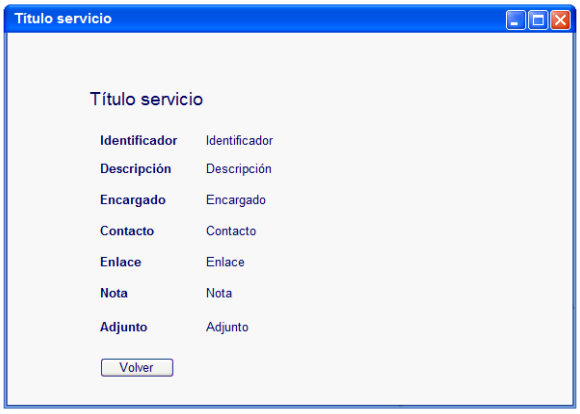
- ConexionLdap: para la autenticación de los usuarios es necesario recoger la CUASI de cada usuario y comprobar la autenticación contra el LDAP. Esta clase contiene un único método, **ConectaLDAP**, que se conecta al LDAP de la Universidad.

Las nuevas clases y métodos del paquete negocio son:

- Restriccion: esta clase, aparte de los métodos para acceder y modificar los atributos de la clase contiene los métodos necesarios para acceder desde la parte de presentación a la parte de datos. Los métodos de esta clase son:
    - **obtenerRestricciones**: llama al método obtenerRestricciones de RestriccionesBD.
    - **nuevaRestriccion**: recoge los parámetros de la página jsp llama al método insertarRestriccion.
    - **borrarRestriccion**: recoge los datos de la restricción que se desea borrar y llama al método borrarRestriccion.
    - **tienePermiso**: llama al método tienePermiso de RestriccionesBD.

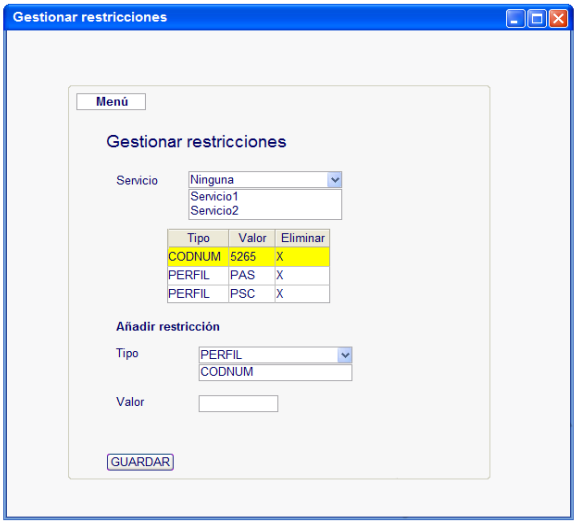
6.2.3 Diseño de interfaces.

Servicio



Buscar servicios

Gestionar restricciones



Añadir restricción

6.3 Construcción

6.3.1 Implementación

Tecnologías empleadas

Las tecnologías empleadas en esta fase de la aplicación son las mismas que se han utilizado en las fases anteriores.

Construcción de la base de datos

Como se ha indicado en la parte de análisis, se ha utilizado el mismo esquema Catálogo que además de las tablas que ya tenía, se ha añadido la tabla restricciones.

Aparte de este esquema, ha sido necesario acceder a otras tablas pertenecientes a otra base de datos. Por lo tanto, hemos tenido que dar al usuario Catálogo permisos de acceso a esas tablas que son la tabla de usuarios, necesaria para obtener el identificador único del usuario, otra tabla que nos devuelve todos los perfiles existentes en la universidad y por último, otra tabla que sirve para obtener los perfiles que tiene un usuario, ya que un usuario puede tener varios perfiles y puede ser a la vez PAS y PSC, por ejemplo.

Lógica de la aplicación

Las partes más destacables del código son:

Autenticación de usuarios:

Para la tercera fase de la aplicación los usuarios deben autenticarse para poder acceder a los servicios que tienen restricciones. Para la autenticación se utiliza la CUASI, es decir, el login y la contraseña personal de cada usuario. Como las contraseñas no están almacenadas en base de datos, es necesario utilizar el LDAP para comprobar la correcta autenticación. Para la conexión se necesita un properties con los datos de la conexión:

```
HOST_LDAP = host1, host2, host3
PUERTO_LDAP = 636
BASE_LDAP = OU=Usuarios
USER_LDAP = CN=usuario
PASSWORD_USER = contraseña
```

Una vez creado este properties en el proyecto, se necesita una clase que recoja los datos y mediante el método conectarSeguro() de la librería ur.jar, se conecte al LDAP:

```
public ConexionLDAP conectaLDAP() throws NamingException {
    //Cogemos los parámetros del fichero conexion.properties
    ResourceBundle rb = ResourceBundle.getBundle("conexion");
    String base = rb.getString("BASE_LDAP");
    String puerto = rb.getString("PUERTO_LDAP");
    String usuario = rb.getString("USER_LDAP");
    String password = rb.getString("PASSWORD_USER");
    String datosHost = rb.getString("HOST_LDAP");
    String[] hosts = datosHost.split(";");
    String host = "";
    ConexionLDAP conexion = new ConexionLDAP();
    int i = 0;
    while (i < hosts.length) {
        //Conecta con el directorio
        host = hosts[i];
        conexion.setBase(base);
        conexion.setHost(host, puerto);
        conexion.setBindDN(usuario, password);
        conexion.conectarSeguro();
        if (conexion.conectado()) {
            return conexion;
        }
        i++;
    }

    return conexion;
}
```

Para comprobar si la autenticación de un usuario es correcta, habrá que llamar a la clase anterior, que crea la conexión y luego utilizar el método autentica, del paquete ur.jar:

```
ConexionLdap con = new ConexionLdap();
ConexionLDAP conexAnon = con.conectaLDAP();
UtilLDAP utilLdap = new UtilLDAP(conexAnon);
if (utilLdap.autentica(login, password)) {
```

Interfaces del sistema

Servicio

Buscar servicios

Gestionar restricciones

Añadir restricción

6.3.2 Pruebas

Para facilitar la tarea se han vuelto a realizar las pruebas unitarias y de integración conjuntamente.

Como en las fases anteriores, la capa de presentación se ha probado en los diferentes navegadores.

De nuevo, se ha creado una clase de prueba para comprobar el funcionamiento correcto de los métodos principales.

```
public class prueba {
    public static void main(String[] args) {
        try {
            ServicioBD servicioBD = new ServicioBD();
            RestriccionBD restriccionBD = new RestriccionBD();

            System.out.println("EL SERVICIO ES PÚBLICO? "+servicioBD.esPublico("Servicio2"));
            ArrayList<Restriccion> rest = restriccionBD.obtenerRestricciones("Servicio1");
            for(int i=0; i<rest.size();i++){
                System.out.println("TIPO: "+rest.get(i).getTipoRestriccion());
                System.out.println("VALOR: "+rest.get(i).getValor());
            }

            restriccionBD.insertarRestriccion("Servicio1", "CODNUM", "5555");
            System.out.println("TIENE PERMISO EL USUARIO 1447 EN EL Servicio1? "
                    +restriccionBD.tienePermiso(1447, "Servicio1"));

        } catch (SQLException ex) {
            Logger.getLogger(prueba.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

En esta clase se muestra si el servicio es público o no, se muestran las restricciones de un servicio, se inserta una restricción y se pregunta si un usuario puede acceder o no a un servicio. El resultado de la ejecución de esta clase es el siguiente:

```
run:
EL SERVICIO ES PÚBLICO? true
TIPO: CODNUM
VALOR: 1447
TIPO: Perfil
VALOR: PAS
true
TIENE PERMISO EL USUARIO 1447 EN EL Servicio1? true
BUILD SUCCESSFUL (total time: 0 seconds)
```

Todos los métodos han funcionado correctamente.

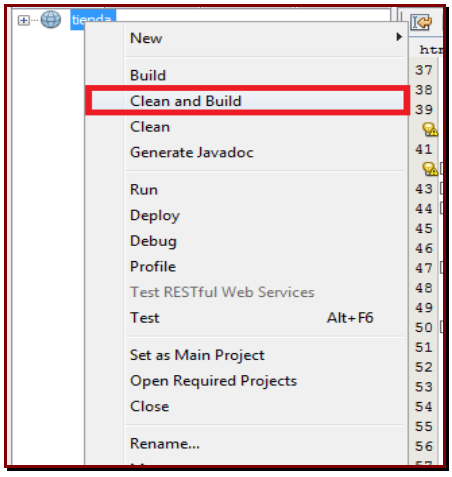
El resto de métodos de la aplicación se han probado haciendo pruebas directamente utilizando la interfaz de la aplicación.

6.3.3 Despliegue

Para desplegar la aplicación en local, basta con arrancar el servidor Apache Tomcat y la aplicación con la estructura de directorios que nos genera el NetBeans. Cuando se ha trabajado en local, se ha utilizado la base de datos de desarrollo, cuya dirección de acceso se ha indicado en el properties GruposConexionesBD:

```
# Desarrollo
bd.url=jdbc:URL:desarrollo
```

Sin embargo para desplegar la aplicación en un servidor, tenemos que generar un archivo war. Este archivo lo podemos generar desde el NetBeans, haciendo clic con el botón derecho sobre el proyecto y seleccionando la opción Clean and Build



Una vez generado el war, bastará con subirlo al servidor. Los servidores del Servicio Informático para este tipo de aplicaciones son Apache Tomcat por lo que habrá que cargar la aplicación y cambiar las direcciones de acceso a la base de datos ya que una vez subida la aplicación habrá que acceder a la base de datos de producción:

```
#Produccion
bd.url=jdbc:URL:produccion
```

7. CONCLUSIONES

Comparativa de planificaciones:

Estaba previsto terminar el proyecto el 21-05-2013. La finalización real ha sido el 7-06-2013. Este pequeño retraso se ha debido a la dificultad en algunas partes del desarrollo de la aplicación, que ha hecho que tuviera que dedicarle más tiempo a esas partes que el que se había estimado y también a que durante la realización del proyecto estaba cursando las asignaturas que me faltaban para acabar el grado y por lo tanto, no he podido dedicarle todo el tiempo que había estimado.

Este retraso no provoca ningún problema ya que seguimos dentro de los plazos de entrega del proyecto.

Conclusiones finales:

Como no era la primera vez que realizaba una aplicación para el Servicio Informático, ha sido sencillo tratar con ellos y fijar los objetivos porque ya tenía los conocimientos suficientes para saber lo que querían y el aspecto que debería tener la aplicación.

Por otra parte, es una experiencia gratificante el haber trabajado con un cliente real y haber tenido que preparar reuniones reales.

En cuanto a los conocimientos que he adquirido al realizar esta aplicación, aunque Java es un lenguaje que manejaba correctamente, me ha servido para reforzar esos conocimientos y aprender sobre nuevas librerías. También el haber utilizado la base de datos Oracle me ha servido para tener más contacto con este gestor de bases de datos. El haber tenido que trabajar con el LDAP es enriquecedor ya que no se tiene siempre la oportunidad de trabajar con este tipo de directorios.

Anexo I

ACTAS DE REUNIÓN

Contenido

Acta reunión I: 51
Acta reunión II: 542
Acta reunión III: 553
Acta reunión IV: 564
Acta reunión V: 575
Acta reunión VI: 586
Acta reunion VII: 597
Acta reunión VIII: 58
Acta reunión IX: 59
Acta reunión X: 60
Acta reunión XI: 61

	Catálogo de servicios	
	Acta de reunión	Servicio Informático

Proyecto:	Diseño y desarrollo de un gestor de "Catálogo de Servicios"
Fecha:	24-01-2013
Lugar:	Servicio Informático – Universidad de La Rioja
Hora inicio:	10:00
Hora finalización:	12:00
Asistentes:	**Asistente** / **Empresa** Txus Cortabarría Giménez / Servicio Informático María Ibeas Ochoa
Orden del día:	Toma de contacto con el proyecto
Documentos relacionados:	Documento con los requisitos iniciales del proyecto y la información que debe contener la base de datos.

Temas tratados:

Indicaciones de los recursos a los que se tendrá acceso durante la realización del proyecto.

Presentación del lugar de trabajo y preparación del equipo.

Explicación del proyecto que se va a llevar a cabo.

Objetivo del proyecto, requisitos y control de acceso.

Temas pendientes

A continuación se listan los temas que quedan pendientes de realizar, los responsables de hacerlo y la fecha prevista para ello.

Tema	Responsables	Fecha resolución
Realización del DOP	María Ibeas	

	Catálogo de servicios	
	Acta de reunión	Servicio Informático

Proyecto:	Diseño y desarrollo de un gestor de “Catálogo de Servicios”
Fecha:	05-02-2013
Lugar:	Servicio Informático – Universidad de La Rioja
Hora inicio:	13:30
Hora finalización:	14:00
Asistentes:	**Asistente** — **Empresa** Juan José Olarte — Universidad de La Rioja María Ibeas Ochoa
Orden del día:	Revisión de la primera parte del DOP

Temas tratados:

Revisión del primer DOP realizado para la aplicación.

Temas pendientes

A continuación se listan los temas que quedan pendientes de realizar, los responsables de hacerlo y la fecha prevista para ello.

Tema	Responsables	Fecha resolución
Corrección del DOP	María Ibeas	07-02-2013

	Catálogo de servicios	
	Acta de reunión	Servicio Informático

Proyecto:	Diseño y desarrollo de un gestor de “Catálogo de Servicios”
Fecha:	08-02-2013
Lugar:	Servicio Informático – Universidad de La Rioja
Hora inicio:	11:30
Hora finalización:	12:30
Asistentes:	**Asistente** — **Empresa** Juan José Olarte — Universidad de La Rioja María Ibeas Ochoa
Orden del día:	Revisión del DOP finalizado

Temas tratados:

Revisión del DOP finalizado.

Damos por finalizado el Documento de Objetivos del Proyecto.

	Catálogo de servicios	
	Acta de reunión	Servicio Informático

Proyecto:	Diseño y desarrollo de un gestor de “Catálogo de Servicios”
Fecha:	11-02-2013
Lugar:	Servicio Informático – Universidad de La Rioja
Hora inicio:	12:00
Hora finalización:	12:30
Asistentes:	**Asistente** / **Empresa** Txus Cortabarría Giménez / Servicio Informático María Ibeas Ochoa
Orden del día:	Diseño de la base de datos

Temas tratados:

Explicación de la base de datos que se usará en el proyecto.

Creación del usuario necesario para acceder a la base de datos.

Asignación de los permisos necesarios para el acceso a la BD.

Explicación de dudas referentes a las categorías y los servicios.

Temas pendientes

A continuación se listan los temas que quedan pendientes de realizar, los responsables de hacerlo y la fecha prevista para ello.

Tema	Responsables	Fecha resolución
Realización prototipos de interfaz	María Ibeas	18-02-2013

	Catálogo de servicios	
	Acta de reunión	Servicio Informático

Proyecto:	Diseño y desarrollo de un gestor de “Catálogo de Servicios”
Fecha:	20-02-2013
Lugar:	Servicio Informático – Universidad de La Rioja
Hora inicio:	11:30
Hora finalización:	12:00
Asistentes:	**Asistente** — **Empresa** Txus Cortabarría Giménez — Servicio Informático María Ibeas Ochoa
Orden del día:	Enseñar al cliente los prototipos de interfaz

Temas tratados:

Mostrar al cliente los prototipos de interfaz de la primera fase de la aplicación para obtener el visto bueno y poder continuar con la aplicación.

Temas pendientes

A continuación se listan los temas que quedan pendientes de realizar, los responsables de hacerlo y la fecha prevista para ello.

Tema	Responsables	Fecha resolución
Finalización fase 1	María Ibeas	11-03-2013

	Catálogo de servicios	
	Acta de reunión	Servicio Informático

Proyecto:	Diseño y desarrollo de un gestor de "Catálogo de Servicios"
Fecha:	15-03-2013
Lugar:	Servicio Informático – Universidad de La Rioja
Hora inicio:	12:00
Hora finalización:	12:30
Asistentes:	**Asistente** / **Empresa** Txus Cortabarría Giménez / Servicio Informático María Ibeas Ochoa
Orden del día:	Enseñar al cliente la fase 1 de la aplicación

Temas tratados:

Se ha enseñado al cliente todas funcionalidades de la primera fase de la aplicación.

Creación de categorías.

Creación de servicios.

Consulta de servicios y categorías.

Temas pendientes

A continuación se listan los temas que quedan pendientes de realizar, los responsables de hacerlo y la fecha prevista para ello.

Tema	Responsables	Fecha resolución
Realización prototipos de interfaz fase 2	María Ibeas	20-03-2013

	Catálogo de servicios	
	Acta de reunión	Servicio Informático

Proyecto:	Diseño y desarrollo de un gestor de “Catálogo de Servicios”
Fecha:	22-03-2013
Lugar:	Servicio Informático – Universidad de La Rioja
Hora inicio:	12:00
Hora finalización:	12:30
Asistentes:	**Asistente** — **Empresa** Txus Cortabarría Giménez — Servicio Informático María Ibeas Ochoa
Orden del día:	Enseñar al cliente los prototipos de interfaz de la fase 2

Temas tratados:

Mostrar al cliente los prototipos de interfaz de la segunda fase de la aplicación para obtener el visto bueno y poder continuar con la aplicación.

Temas pendientes

A continuación se listan los temas que quedan pendientes de realizar, los responsables de hacerlo y la fecha prevista para ello.

Tema	Responsables	Fecha resolución
Finalización fase 2	María Ibeas	12-04-2013

	Catálogo de servicios	
	Acta de reunión	Servicio Informático

Proyecto:	Diseño y desarrollo de un gestor de "Catálogo de Servicios"
Fecha:	08-04-2013
Lugar:	Servicio Informático – Universidad de La Rioja
Hora inicio:	11:00
Hora finalización:	12:30
Asistentes:	**Asistente** / **Empresa** Juan José Olarte / Universidad de La Rioja María Ibeas Ochoa
Orden del día:	Revisar con el tutor la memoria realizada hasta el momento

Temas tratados:

Revisión de las fases de la memoria que se han realizado hasta el momento.

Temas pendientes

A continuación se listan los temas que quedan pendientes de realizar, los responsables de hacerlo y la fecha prevista para ello.

Tema	Responsables	Fecha resolución
Continuar con las fases que faltan de la memoria y realizar las correcciones de la parte revisada.	María Ibeas	20-05-2013

	Catálogo de servicios	
	Acta de reunión	Servicio Informático

Proyecto:	Diseño y desarrollo de un gestor de "Catálogo de Servicios"
Fecha:	20-04-2013
Lugar:	Servicio Informático – Universidad de La Rioja
Hora inicio:	12:00
Hora finalización:	12:30
Asistentes:	**Asistente** / **Empresa** Txus Cortabarría Giménez / Servicio Informático María Ibeas Ochoa
Orden del día:	Enseñar al cliente los prototipos de interfaz de la fase 3

Temas tratados:

Mostrar al cliente los prototipos de interfaz de la tercera fase de la aplicación para obtener el visto bueno y poder continuar con la aplicación.

Temas pendientes

A continuación se listan los temas que quedan pendientes de realizar, los responsables de hacerlo y la fecha prevista para ello.

Tema	Responsables	Fecha resolución
Finalización fase 3	María Ibeas	14-05-2013

	Catálogo de servicios	
	Acta de reunión	Servicio Informático

Proyecto:	Diseño y desarrollo de un gestor de “Catálogo de Servicios”
Fecha:	16-05-2013
Lugar:	Servicio Informático – Universidad de La Rioja
Hora inicio:	10:00
Hora finalización:	11:00
Asistentes:	**Asistente** — **Empresa** Txus Cortabarría Giménez — Servicio Informático María Ibeas Ochoa
Orden del día:	Enseñar al cliente la aplicación completa

Temas tratados:

Se ha enseñado al cliente todas funcionalidades de la tercera fase de la aplicación.

Página de los servicios.

Gestión de restricciones.

Búsqueda de servicios.

	Catálogo de servicios	
	Acta de reunión	Servicio Informático

Proyecto:	Diseño y desarrollo de un gestor de “Catálogo de Servicios”
Fecha:	03-06-2013
Lugar:	Servicio Informático – Universidad de La Rioja
Hora inicio:	11:30
Hora finalización:	13:00
Asistentes:	**Asistente** — **Empresa** Juan José Olarte — Universidad de La Rioja María Ibeas Ochoa
Orden del día:	Revisar con el tutor la memoria completa

Temas tratados:

Se ha enseñado al tutor del proyecto la memoria completa. Se han realizado varias correcciones y se ha dejado la memoria lista para entregar una vez realizadas esas correcciones.

Anexo II

MANUAL DE USUARIO

OBJETIVO 63

ESCENARIO 63

PROCEDIMIENTO 63

Gestionar categorías 63

Nueva categoría 63

Modificar categoría 64

Borrar categoría 64

Gestionar servicios 65

Nuevo servicio 65

Modificar servicio 66

Borrar servicio 67

Gestionar restricciones 67

Añadir y borrar restricciones 67

Ver el catálogo de servicios 68

Ver servicios 68

Buscar servicios 69

OBJETIVO

Explicar las diferentes opciones que proporciona la aplicación del Catálogo de Servicios.

ESCENARIO

Con el objetivo de mostrar un catálogo con todos los servicios que proporciona el Servicio Informático de la Universidad de La Rioja se ha desarrollado una aplicación que permite, entre otras, la consulta y búsqueda de los servicios por parte de los usuarios visitantes, y la gestión de los servicios, categorías y restricciones de las categorías y los servicios por parte de los usuarios administradores.

PROCEDIMIENTO

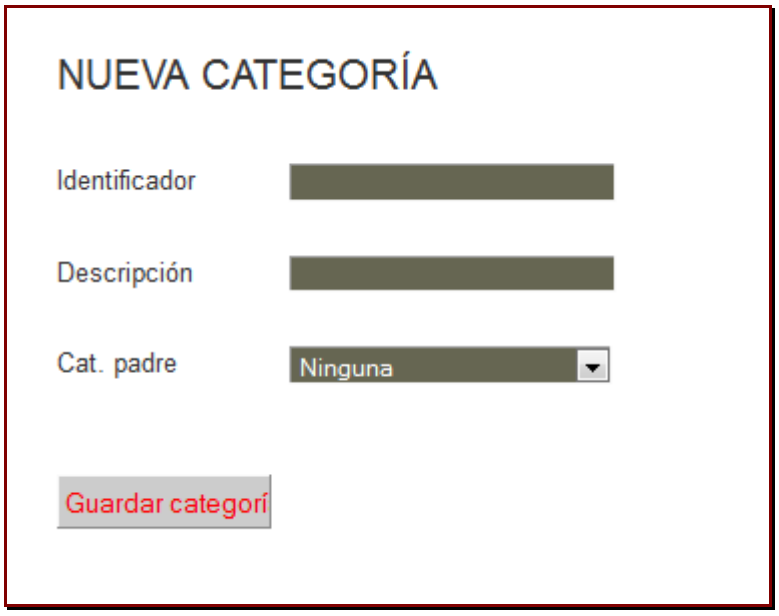
El usuario administrador dispone de las siguientes opciones:

Gestionar categorías

El administrador podrá crear, modificar y borrar las categorías que luego se mostrarán en el catálogo de servicios.

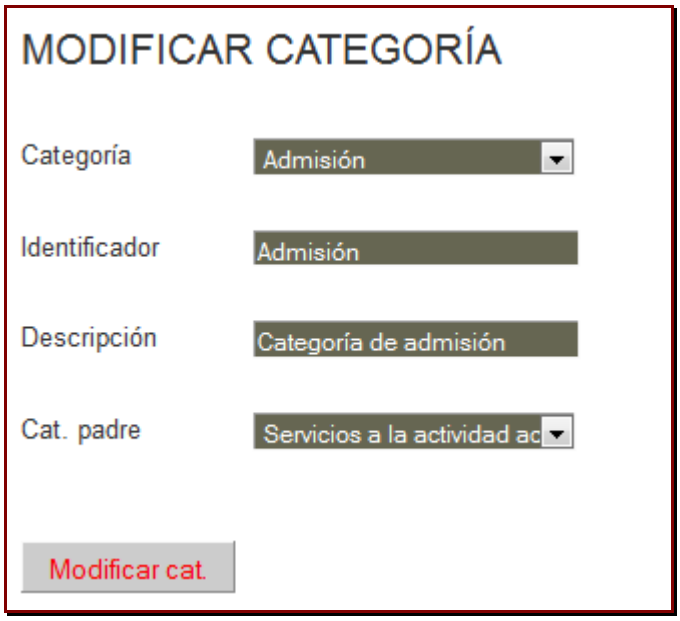
Nueva categoría

Para crear una nueva categoría el administrador deberá rellenar los campos del formulario que se muestra a continuación indicando el identificador de la categoría, la descripción de la categoría y, si tuviera, la categoría padre.



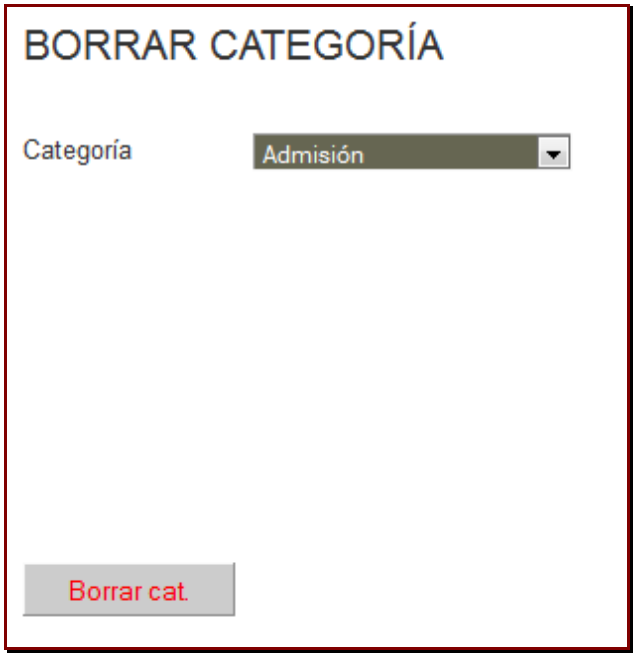
Modificar categoría

El administrador podrá modificar las categorías existentes. Para ello sólo tendrá que seleccionar del combo de categorías la categoría que desee modificar y automáticamente se cargarán los datos de esa categoría. El administrador únicamente tendrá que modificar los valores que desee.



Borrar categoría

El administrador podrá borrar cualquier categoría existente. De la misma forma que en el caso anterior, únicamente tendrá que seleccionar del combo la categoría que desee borrar y hacer clic en el botón de borrar categoría.

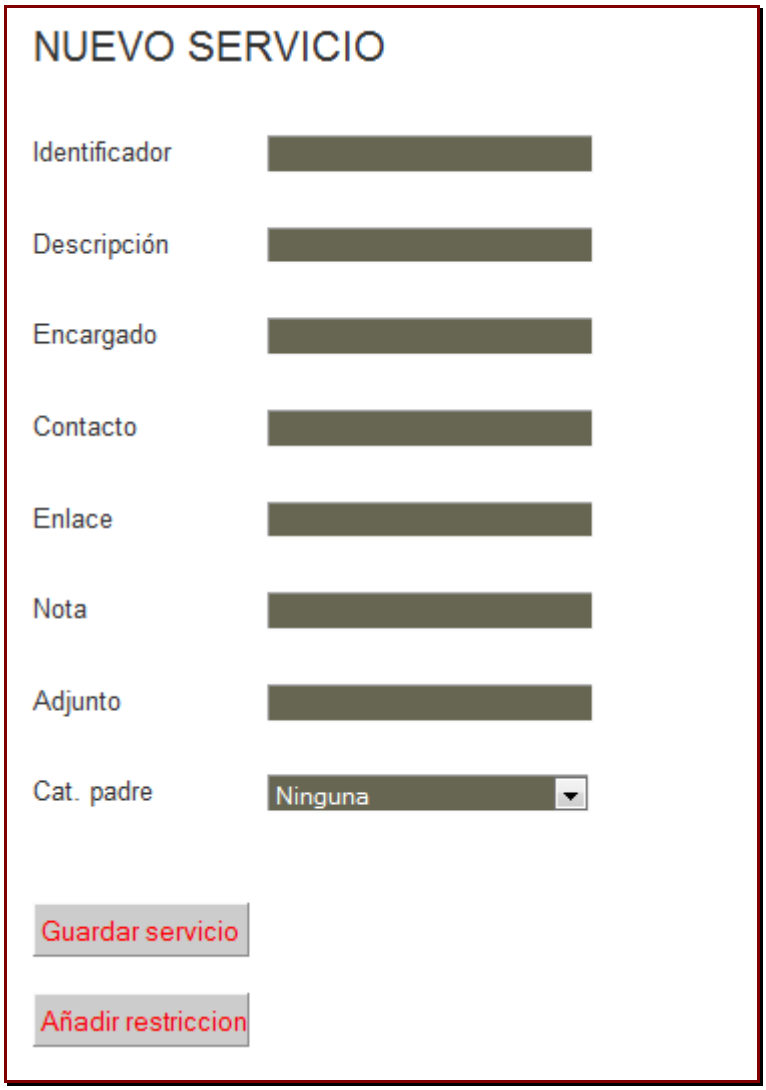


Gestionar servicios

El administrador de la aplicación podrá crear nuevos servicios, o modificar y borrar los servicios existentes.

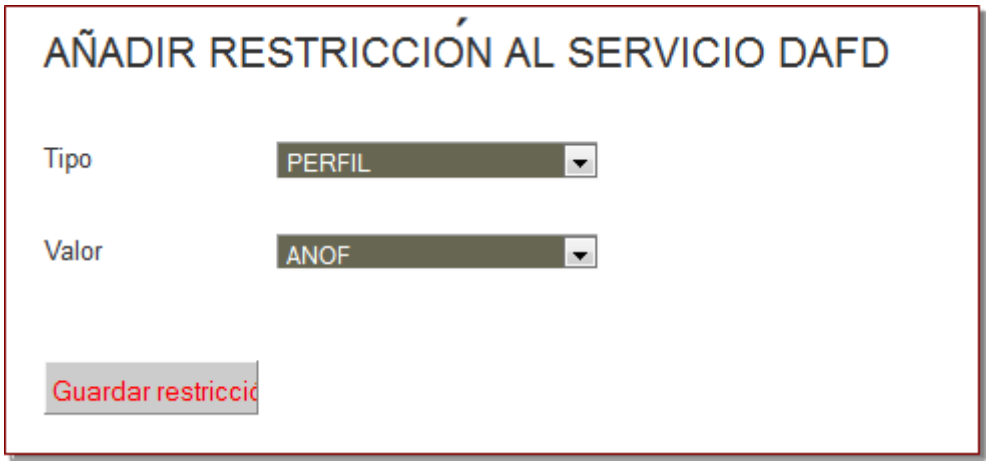
Nuevo servicio

El administrador podrá crear nuevos servicios rellenando el siguiente formulario donde deberá indicar datos como el identificador, la descripción del servicio y la categoría a la que pertenece el servicio.



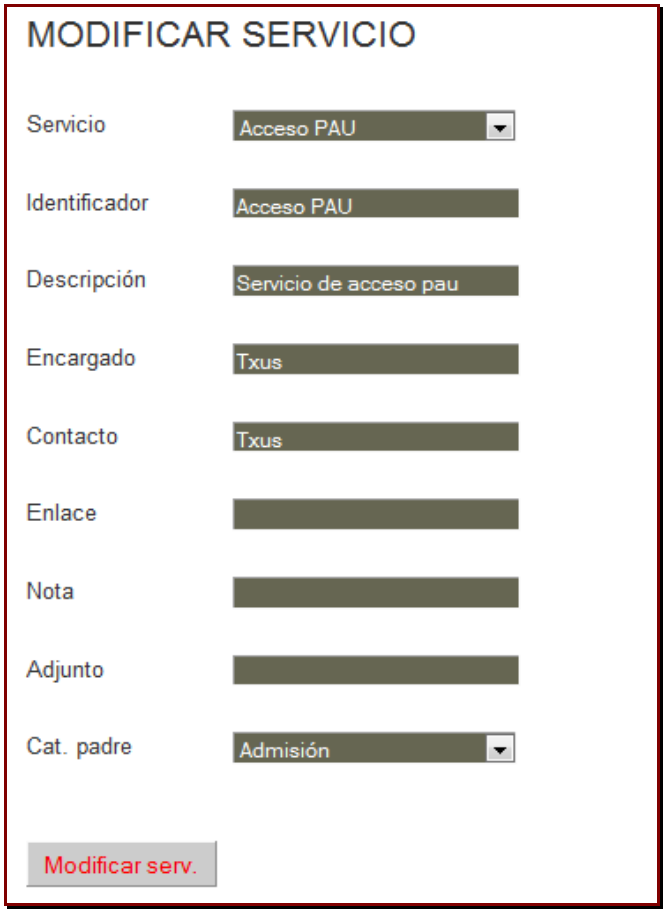
Una vez rellenado el formulario, el administrador podrá poner el servicio a público y por lo tanto hacer clic en Guardar servicio, o añadir restricciones de acceso y por lo tanto hacer clic en Añadir restricción. De esta forma el administrador podrá añadir restricciones de acceso al servicio.

Estas restricciones pueden ser por perfil o por usuario.



Modificar servicio

El administrador podrá modificar los servicios creados. Al igual que con las categorías, bastará con seleccionar un servicio del combo que contiene todos los servicios, y al seleccionarlo se cargarán todos los datos del servicio. El administrador únicamente tendrá que cambiar los datos que desee del servicio:



Borrar servicio

El administrador podrá también borrar los servicios creados. Como en el caso anterior, bastará con seleccionar de la lista el servicio que desee borrar y hacer clic en el botón de borrar servicio:

Gestionar restricciones

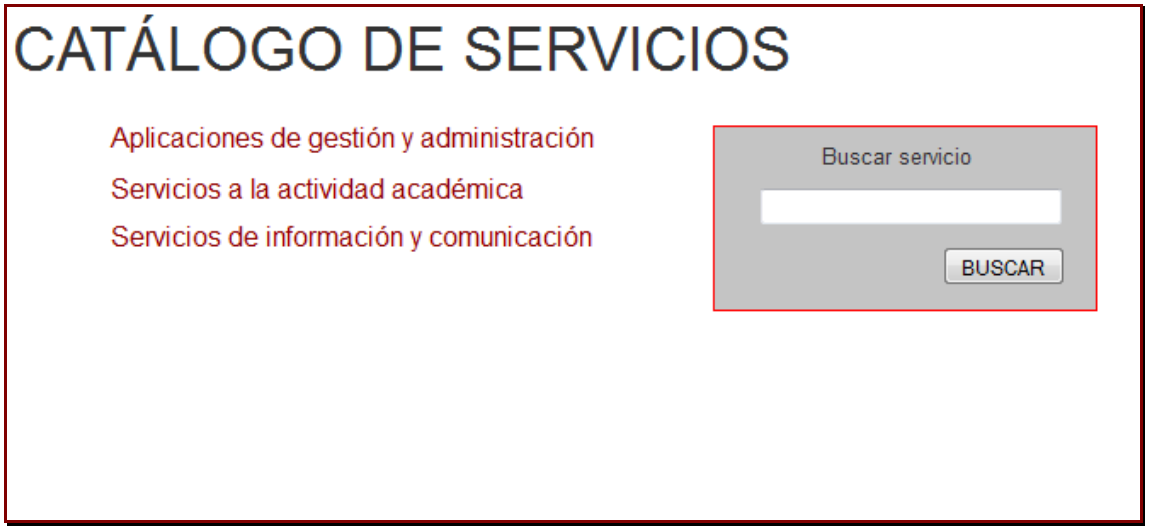
Añadir y borrar restricciones

Por último el administrador podrá gestionar el acceso de los servicios añadiendo o borrando restricciones. Bastará con seleccionar el servicio y si desea añadir nuevas restricciones deberá rellenar el formulario y hacer clic en Crear restricción, y si desea borrar alguna de las restricciones deberá hacer clic en el botón eliminar de la restricción que quiera borrar:

El usuario visitante dispone de las siguientes opciones:

Ver el catálogo de servicios

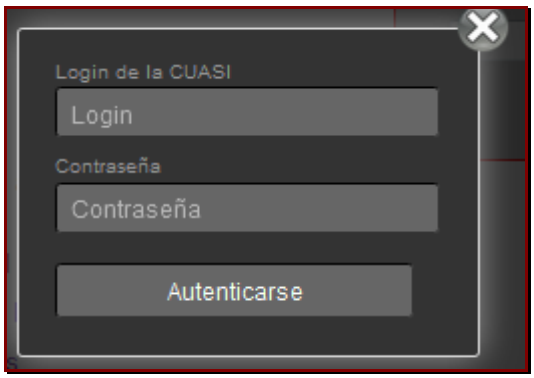
Los visitantes podrán consultar el catálogo de servicios que consiste en una lista que al hacer clic en las categorías se despliega.



Ver servicios

Cuando un usuario hace clic en un servicio, si ese servicio es público, le mostrará directamente la página con la información de ese servicio.

En cambio, si el servicio tiene restricciones de acceso, al hacer clic en el servicio, se le mostrará al usuario el siguiente cuadro de autenticación:

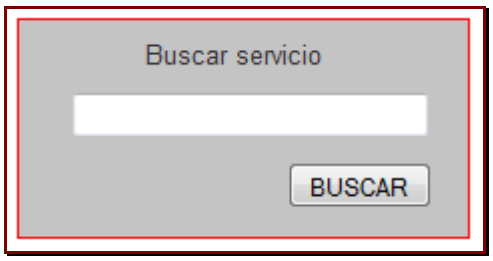


El usuario deberá indicar su login y contraseña de la CUASI. Si la autenticación del usuario es correcta, y el usuario tiene permisos de acceso a ese servicio, se le mostrará la información del servicio.

Sin embargo, si el usuario no se ha autenticado correctamente, se le mostrará el siguiente cuadro de error.

Buscar servicios

Los usuarios podrán buscar servicios introduciendo palabras clave en el buscador:



El resultado de la búsqueda se mostrará en forma de listado: